The Hypergraph Assignment Problem

vorgelegt von
Olga Heismann, M. Sc.
aus Nikolaew

von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. John M. Sullivan
Gutachter:	Prof. Dr. Ralf Borndörfer
Gutachter:	Prof. Dr. Dr. h. c. mult. Martin Grötschel

Tag der wissenschaftlichen Aussprache: 27.05.2014

Berlin 2014
D 83

Zusammenfassung

Diese Arbeit beschäftigt sich mit dem Hypergraph Assignment Problem (Abkürzung „HAP“, dt.: Zuordnungsproblem auf Hypergraphen), einem Mengenzerlegungsproblem auf einem speziellen Typ von Hypergraphen. Das HAP verallgemeinert das Zuordnungsproblem von bipartiten Graphen auf eine Struktur, die wir bipartite Hypergraphen nennen, und ist durch eine Anwendung in der Umlaufplanung im Schienenverkehr motiviert. Die Hauptresultate betreffen die Komplexität, polyedrische Ergebnisse, die Analyse von Zufallsinstanzen sowie primale Methoden für das HAP.

Wir beweisen, dass das HAP $\mathcal{NP}$-schwer und $\mathcal{APX}$-schwer ist, sogar wenn wir uns auf kleine Hyperkantengrößen und Hypergraphen mit einer speziellen, partitionierten Struktur beschränken. Darüber hinaus untersuchen wir die Komplexität der Mengenpackungs- sowie Mengenüberdeckungsrelaxierung und geben für bestimmte Fälle Approximations- und exakte Algorithmen mit einer polynomiellen Laufzeit an.

Für das Polytop des Zuordnungsproblems ist eine vollständige lineare Beschreibung bekannt. Wir untersuchen daher auch das HAP-Polytop. Dafür ist die Anzahl der Facettenungleichungen schon für sehr kleine Problemgrößen sehr groß. Wir beschreiben eine Methode zur Aufteilung der Ungleichungen in Äquivalenzklassen, die ohne die Verwendung von Normalformen auskommt. Die Facetten in jeder Klasse können durch Symmetrien ineinander überführt werden. Es genügt, einen Repräsentanten aus jeder Klasse anzugeben, um ein vollständiges Bild der Polytopstruktur zu erhalten. Wir beschreiben den Algorithmus „HUHFA“, der diese Klassifikation nicht nur für das HAP, sondern für beliebige kombinatorische Optimierungsprobleme, die Symmetrien enthalten, durchführt.

Die größtmögliche HAP-Instanz, für die wir die vollständige lineare Beschreibung berechnen konnten, hat 14049 Facetten, die in 30 Symmetrieklassen aufgeteilt werden können. Wir können 16 dieser Klassen kombinatorisch interpretieren. Dafür verallgemeinern wir Odd-Set-Ungleichungen für das Matchingproblem unter Verwendung von Cliquen. Die Ungleichungen, die wir erhalten, sind gültig für Mengenpackungsprobleme in beliebigen Hypergraphen und haben eine klare kombinatorische Bedeutung.

Die Analyse von Zufallsinstanzen erlaubt einen besseren Einblick in die Struktur von Hyperzuordnungen. Eine solche ausführliche Analyse wurde in der Literatur theoretisch und praktisch bereits für das Zuordnungsproblem durchgeführt. Als eine Verallgemeinerung dieser Ergebnisse für das HAP beweisen wir Schranken für den Erwartungswert einer Hyperzuordnung mit minimalen Kosten, die genau die Hälfte der maximal möglichen Anzahl an Hyperkanten,

die keine Kanten sind, benutzt. In einem sog. vollständigen partitionierten Hypergraphen $G_{2,2n}$ mit Hyperkantenkosten, die durch unabhängig identisch exponentiell verteilte Zufallsvariablen mit Erwartungswert 1 bestimmt sind, liegt dieser Wert zwischen 0.3718 und 1.8310, wenn die Knotenanzahl gegen unendlich strebt.

Schließlich entwickeln wir eine exakte kombinatorische Lösungsmethode für das HAP, die drei Ansätze kombiniert: Eine Nachbarschaftssuche mit Nachbarschaften exponentieller Größe, die Composite-Columns-Methode für das Mengenzerlegungsproblem sowie den Netzwerksimplexalgorithmus.

Abstract

This thesis deals with the hypergraph assignment problem (HAP), a set partitioning problem in a special type of hypergraph. The HAP generalizes the assignment problem from bipartite graphs to what we call bipartite hypergraphs, and is motivated by applications in railway vehicle rotation planning. The main contributions of this thesis concern complexity, polyhedral results, analyses of random instances, and primal methods for the HAP.

We prove that the HAP is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard even for small hyperedge sizes and hypergraphs with a special partitioned structure. We also study the complexity of the set packing and covering relaxations of the HAP, and present for certain cases polynomial exact or approximation algorithms.

A complete linear description is known for the assignment problem. We therefore also study the HAP polytope. There, we have a huge number of facet-defining inequalities already for a very small problem size. We describe a method for dividing the inequalities into equivalence classes without resorting to a normal form. Within each class, facets are related by certain symmetries and it is sufficient to list one representative of each class to give a complete picture of the structural properties of the polytope. We propose the algorithm "HUHFA" for the classification that is applicable not only to the HAP but combinatorial optimization problems involving symmetries in general.

In the largest possible HAP instance for which we could calculate the complete linear description, we have 14049 facets, which can be divided into 30 symmetry classes. We can combinatorially interpret 16 of these classes. This is possible by employing cliques to generalize the odd set inequalities for the matching problem. The resulting inequalities are valid for the polytope associated with the set packing problem in arbitrary hypergraphs and have a clear combinatorial meaning.

An analysis of random instances provides a better insight into the structure of hyperassignments. Previous work has extensively analyzed random instances for the assignment problem theoretically and practically. As a generalization of these results for the HAP, we prove bounds on the expected value of a minimum cost hyperassignment that uses half of the maximum possible number of hyperedges that are not edges. In a certain complete partitioned hypergraph $G_{2,2n}$ with i. i. d. exponential random variables with mean 1 as hyperedge costs it lies between 0.3718 and 1.8310 if the vertex number tends to infinity.

Finally, we develop an exact combinatorial solution algorithm for the HAP that combines three methods: A very large-scale neighborhood search, the composite columns method for the set partitioning problem, and the network simplex algorithm.

Acknowledgments

All my work on the subject of this thesis was done during my time at Zuse Institute Berlin (ZIB), and I would like to thank Martin Grötschel and Ralf Borndörfer for the possibility to work here. I am very grateful to Ralf Borndörfer also for his support, his unlimited number of ideas, and for teaching me a lot about writing mathematics. Furthermore, my time at ZIB would have been very different without my colleagues, especially without those with whom I shared so many enjoyable lunch breaks with interesting conversations.

I would like to thank my friend and collaborator Achim Hildenbrandt from the University of Heidelberg for the many inspiring and motivating mathematical discussions we had, and the work we have done together in the last years.

Also, my thanks go to Isabel Beckenbach, Achim Hildenbrandt, Linus Mattauch, Sandra de Ruijter, and Edo Schinzinger for critically reading (parts of) this thesis and their very valuable comments.

Last but not least, I would like to express my gratitude to my parents for showing me at a very young age how mathematics can be fun. One might consider this the very first step that brought this thesis into existence.

The research for this thesis was conducted within the project “Rolling stock roster planning for railways” supported by the DFG Research Center MATHEON.

Preface

Consider the following problem: What is the minimum cost of a covering of the letters A–L with a subset of pairwise disjoint sets ("hyperedges") from the set

$$\big\{\{A,B,D,G,H,J\},\{A,H\},\{A,J\},\{B,G\},\{B,L\},\{C,D,K,L\},\\ \{C,I\},\{D,I\},\{D,J\},\{D,K\},\{E,F,G,H\},\{E,F,I,J\},\\ \{E,I\},\{E,J\},\{F,G\},\{F,J\},\{F,K\},\{F,L\}\big\}$$

if their costs are -100, 0.24, 0.43, 0.13, 0.02, 0.02, 0.19, 0.05, 0.11, 0.81, 0.71, 0.62, 0.04, 0.06, 0.14, 0.53, 0.08, 0.04, respectively? Can you at least prove that no solution with cost ≤ 0 exists for this example of a set partitioning problem?

Figure 1 can! It shows that the question can be represented as a bipartite perfect matching problem, also called an assignment problem, with some of the edges glued together. This allows us to apply Hall's theorem to immediately prove that the problem does not have a solution with cost ≤ 0, i. e., a solution that contains the set $\{A,B,D,G,H,J\}$. Hall's theorem [Hall, 1935] provides the following necessary and sufficient condition for the existence of an assignment. An assignment exists if and only if for each subset of vertices from "one side" of the graph, the number of vertices on the "other side" of the graph to which they are connected by some edge is not less than them. If we select the hyperedge $\{A,B,D,G,H,J\}$, all other hyperedges that can be used (those that do not cover one of the vertices A, B, D, G, H, J) are edges, and the condition from Hall's theorem is violated for the set $\{K,L\}$ in the remaining assignment problem, see Figure 2.

So we know that no solution with the hyperedge $\{A,B,D,G,H,J\}$ exists. But how can we then find an optimal solution? We now discuss by means of this example the primal and dual methods that can be applied to such a type of set partitioning problem with an assignment-like structure, which is the type of problem we deal with in this thesis.

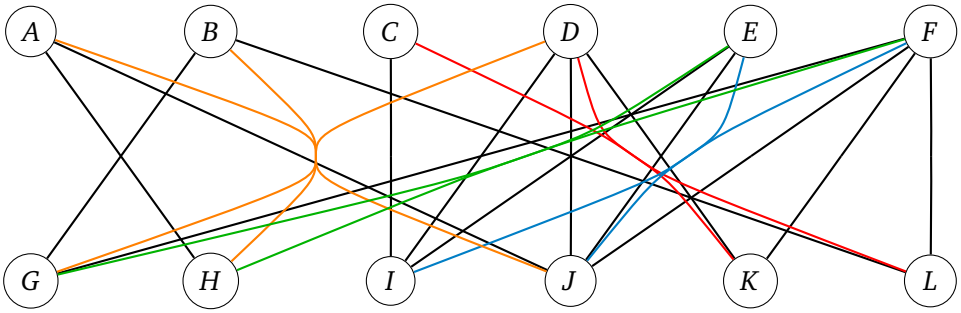


Figure 1: The letters A–L are represented as vertices, the sets are drawn as connections of the corresponding vertices. The vertices are divided into two sides (drawn in the upper and lower area of this figure), and each hyperedge connects the same number of vertices from both sides. The hyperedges that connect only one vertex from each side are also called edges, and the other hyperedges can be viewed as a combination of edges.

For the primal approach, we can make use of this structure to develop a local search heuristic. To find a start solution for the local search, we can restrict ourselves to the sets of cardinality two, the black edges in Figure 1. Then the problem becomes an assignment problem and can be solved in polynomial time. The optimal assignment is $\{\{A,H\},\{B,L\},\{C,I\},\{D,K\},\{E,J\},\{F,G\}\}$, and has cost $0.24+0.02+0.19+0.81+0.06+0.14=1.46$. If not enough edges are present in the problem, one can add edges with very high cost to get some assignment as a start solution.

As the foundation of the local search, we now describe a way to group feasible solutions of this special type of set partitioning problem such that a solution with lowest cost in each group can be found in polynomial time. All assignments, i. e., solutions that consist only of edges, belong to one group. It can be described as follows. For an assignment, the intersections of all the hyperedges in the solution with $\{A,\ldots,F\}$ and $\{G,\ldots,L\}$ are the sets $\{A\},\ldots,\{F\}$ and $\{G\},\ldots,\{L\}$, respectively.

For other solutions, these hyperedge intersections with $\{A,\ldots,F\}$ and $\{G,\ldots,L\}$ are different. If we fix the intersection of the hyperedges in a solution that we are looking for with $\{A,\ldots,F\}$ and $\{G,\ldots,L\}$, only some of the hyperedges can be used, and such a solution with minimum cost can be found in polynomial time. This restricted problem can be viewed as an assignment problem again: If we glue together all the vertices in each hyperedge intersection set, then all hyperedges become edges.

The local search is as follows. In each step, it unites two of the hyper-

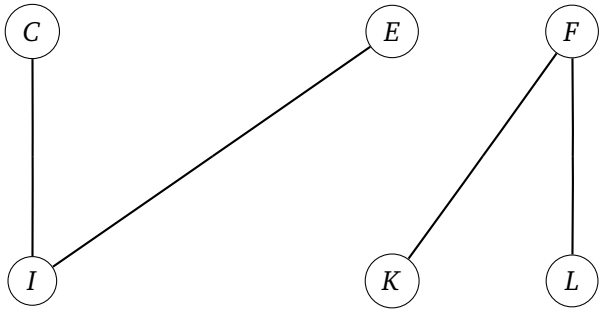


Figure 2: All sets that do not cover one of the vertices A, B, D, G, H, J. Hall's theorem implies that no solution using the set $\{A,B,D,G,H,J\}$ with cost -100 exists, all other hyperedges have a positive cost.

edge intersections with $\{A,\dots,F\}$ and two of the hyperedge intersections with $\{G,\dots,L\}$, and finds a solution with minimum cost with such intersections. From all possibilities to choose two subsets of $\{A,\dots,F\}$ and two subsets of $\{G,\dots,L\}$, the heuristic will take the one where the minimum cost solution has the smallest cost. If such a step—or a step in the other direction, i. e., a subdivision of two such sets—does not lead to a solution with a better cost than the previous solution, the local search has found a local minimum.

For our example, the heuristic will be in a local minimum after just one step. The intersection sets then are $\{\{A\},\{B\},\{C,D\},\{E\},\{F\}\}$ and $\{\{G\},\{H\},\{I\},\{J\},\{K,L\}\}$. The solution in the local minimum is $\{\{A,H\},\{B,G\},\{C,D,K,L\},\{E,I\},\{F,J\}\}$, see Figure 3, with cost $0.24+0.13+0.02+0.04+0.53=0.96$. In Chapter 7, we will show how we can use a combinatorial primal algorithm to escape from a local minimum or prove that it is global. Here, we want to show that the solution found is optimal using dual methods, namely, two types of cut inequalities.

The standard integer linear programming formulation for the problem has one 0/1-variable for each hyperedge, and a constraint for each vertex enforcing exactly one hyperedge that covers the vertex in a feasible solution. If we solve its linear programming relaxation for our example, the solution will not be integral. It is shown in Figure 4, and has cost 0.615. It is easy to see how to separate this fractional solution. The hyperedges $\{C,D,K,L\}$, $\{C,I\}$, $\{D,I\}$ have pairwise a non-empty intersection. Therefore, at most one of them can be part of a solution and the sum of the corresponding variables has to be at most one. Such an additional constraint is called a clique inequality. We will show in Section 6.2 an extended formulation of polynomial size that implies all clique inequalities.

If we add the clique inequality to the LP (or use the extended formulation),

x

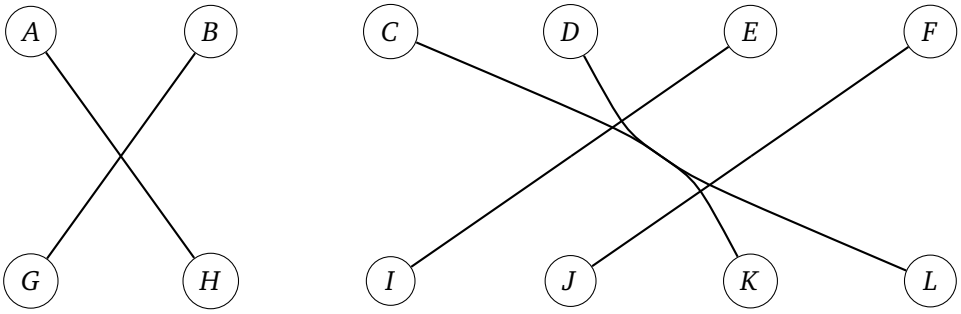


Figure 3: Local minimum of the heuristic.

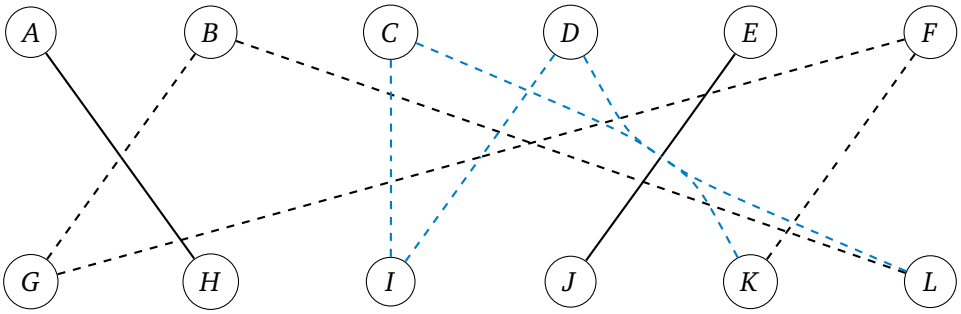


Figure 4: Solution of the LP relaxation. The solid hyperedges have value 1, the dashed ones have value 0.5. The clique inequality that enforces the sum of the variables associated with the blue hyperedges to be at most 1, separates this fractional solution.

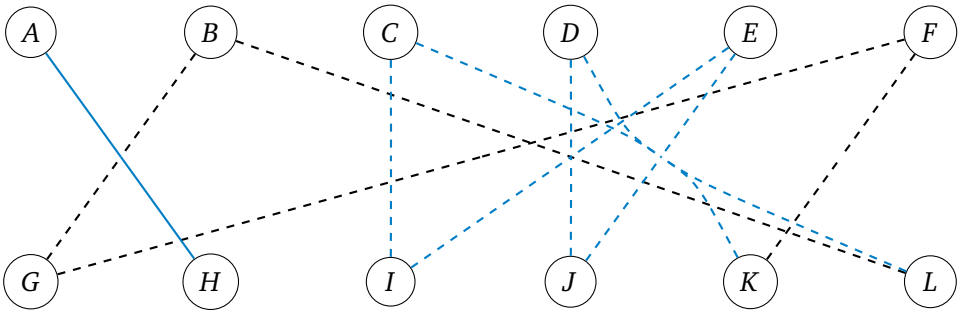


Figure 5: Solution of the LP relaxation after adding a clique inequality. The solid hyperedges have value 1, the dashed ones have value 0.5. The valid inequality that enforces the sum of the variables associated with the blue hyperedges to be at most 3, separates this fractional solution.

the new optimal LP solution will also be fractional. It is shown in Figure 5 and has cost 0.635. Another method is needed to separate this solution. What can achieve the desired separation is a generalization of the odd set inequalities for the matching problem as will be presented in Section 6.3. For our example, we will take the six sets of hyperedges that contain the vertices A, C, D, H, I, J, respectively, and the clique $\{\{E,F,I,J\},\{E,I\},\{E,J\}\}$. From each of the seven sets, at most one hyperedge can be contained in a solution. By a rounding argument, the sum of all variables for which the associated hyperedges are contained in at least two such sets is at most three. This cut separates the previously found fractional solution, and after adding this cut to the LP the optimal LP value is 0.96. This proves that the local minimum of our heuristic is, indeed, an optimal solution.

Could we have guessed the optimal value? In general, this is of course not possible. However, for cost functions drawn from certain distributions one can make predictions about the optimal value. This will be the subject of Chapter 4.

The example discussed above is a set partitioning problem. However, this example is special in the sense that it has an assignment-like structure. We call a set partitioning problem with such a structure the "hypergraph assignment problem" and use the abbreviation "HAP". As already shown for the example, the aim of this thesis is to explore how results from combinatorial optimization problems on graphs such as assignment, matching, or flow problems can be transferred to the hypergraph setting in the HAP. The HAP is provably $\mathcal{NP}$-hard even for bipartite hypergraphs with a very simple structure. Therefore, we cannot expect to obtain a polynomial time algorithm, or get a complete polynomial size representation of the polytope of feasible solutions. We can, however, employ several results for combinatorial optimization problems on graphs to prove results or develop solution methods for the HAP.

The (linear) assignment problem, which is generalized by the HAP, is one of the best known and simultaneously best understood problems in combinatorial optimization. It consists of finding a minimum cost perfect matching, also called an assignment, in a bipartite graph with two equally sized vertex sets and given costs for all the edges. Theoretically efficient and practically fast algorithms, which allow to tackle even very large instances in a very short time, have been developed [Burkard et al., 2012]. This is important since the assignment problem appears in many practical applications, for instance, personnel planning or vehicle assignment. The polytope of the standard integer linear programming formulation for the assignment problem is well-understood. For random instances, expected values and other results are known for certain cost functions [Krokhmal and Pardalos, 2009]. Also, several, often $\mathcal{NP}$-complete, generalizations of the assignment problem such as the quadratic and the multi-

dimensional assignment problem have been investigated. For a survey on the assignment problem and its generalizations, see [Burkard et al., 2012]. For the set partitioning problem, of which the HAP is a special case, results such as polynomial algorithms are not known and will probably never be found as it is $\mathcal{NP}$-hard.

Our generalization of the assignment problem is a hypergraph generalization which sticks to the linear bipartite setting but replaces edges with hyperedges, as shown in the example. In the HAP, each hyperedge connects the same number of vertices from the two vertex sets of the hypergraph. A different assignment-type problem on such a structure that has been investigated before is the hospitals/residents problem with couples [McDermid and Manlove, 2010]. It is a generalization of the stable marriage problem [Gale and Shapley, 1962], i. e., a problem in which not the cost but the stability of the assignment is focused on.

The hypergraph assignment problem is an idealized case of a model for an application in rail transport, more specifically, for vehicle rotation planning for long distance passenger railways. It deals with the allocation of vehicles to trips in a timetable, see [Maróti, 2006]. A vehicle rotation plan can be viewed as an assignment of each trip to a follow-on trip which will be serviced by the same vehicle. In practice, several side constraints such as maintenance and train composition have to be taken into account. One type of these constraints is known as regularity. A vehicle rotation plan is considered operationally regular, if many timetabled trips are followed by the same timetabled trips on as many days of the standard week as possible. For example, if trip 4711 is followed by trip 4712 on Monday, this should also be the case on Tuesday, Wednesday, etc. (provided that these trips exist on these days). In practice, most trips appear on almost every day of operation. In other words, the weekly timetable is largely regular, such that there is a good chance to also construct a regular vehicle rotation plan. Regular vehicle rotation plans are easier to communicate and understand than non-regular ones. They standardize operations, increase robustness, and facilitate real-time scheduling. It is therefore essential to include regularity in vehicle rotation planning models. For further details on the hypergraph model for the vehicle rotation planing problem, see [Borndörfer et al., 2011].

What we call regularity is also important in other scheduling problems in transportation, see, for example, [Amberg et al., 2011] for an approach to this issue for public bus transport or [Klabjan et al., 2001] for airline crew scheduling.

This thesis is structured as follows. In Chapter 1, we will introduce the hypergraph assignment problem together with the associated structures. In Chapter 2, we give an overview of related literature. It summarizes results

for partitioning problems on graphs, the set packing, partitioning and covering problems on general and specially structured hypergraphs, and hyperflow.

Then, in Chapter 3, we will discuss general results for the HAP. First of all, we will prove that the HAP is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard. We will then show that the problem can be analyzed in bipartite hypergraphs with a structure which makes them even more similar to bipartite graphs. These special bipartite hypergraphs will be called partitioned hypergraphs. We will also mention polynomially solvable cases of the HAP, which, however, are only possible for very restricted cases. In one of the polynomial cases that we discuss, the HAP is solved by reducing it to a polynomial number of assignment problems. In the other polynomial case discussed, the HAP can transformed to a perfect matching problem, which implies the polynomial solvability.

To foster our understanding of how hyperassignments work, we analyzed random instances of the HAP for different cost functions. In Chapter 4, we will discuss our observations, and prove bounds on the expected value of optimal solutions. This will be achieved by exploiting the assignment-like structure of the bipartite hypergraphs that we studied and using results on the random assignment problem.

In Chapter 5, we introduce the software "HUHFA" that can be used for arbitrary combinatorial optimization problems to understand the facets of the polytope of feasible solutions by classifying them into symmetry classes. We there also state the theory behind it. The results that we obtained with HUHFA for the HAP allow us to understand certain facets.

In Chapter 6, we will therefore deal with a dual approach to the HAP. There, we will first discuss an extended formulation and its projection to the original variables. Besides others, it implies all the clique inequalities. This is made possible by the special structure of partitioned hypergraphs in contrast to general hypergraphs. Then we discuss a new class of valid and, at least sometimes, facet-defining inequalities. These were inspired by our analysis of a HAP polytope for a certain bipartite hypergraph $G_{2,3}$ using HUHFA, and work not only for the HAP but for set packing or set partitioning problems in general. They are a generalization of odd set inequalities for the matching problem.

The last chapter, Chapter 7, deals with primal methods for the HAP. We develop a mainly combinatorial exact solution method that combines a very large-scale neighborhood search with the composite columns method for the set partitioning problem, and the network simplex algorithm.

Parts of this thesis are joint work and have already been published or submitted for publication. The connection to vehicle rotation planning shortly described in this preface is also part of [Heismann and Borndörfer, 2012]. Parts of Chapter 2 as well as Sections 3.1, 3.2, and 6.2 have been accepted for publi-

cation in Discrete Optimization Journal subject to minor modifications on January 22, 2014 (preprint: [Borndörfer and Heismann, 2012]). A slightly modified version of Section 4.1 has been published as [Heismann and Borndörfer, 2013b]. A slightly modified version of Chapter 5 has been submitted to International Journal of Computational Geometry and Applications on September 9, 2013 (preprint: [Heismann, Hildenbrandt, Silvestri, Reinelt, and Borndörfer, 2013]). Parts of Section 6.3 have been accepted for publication in the post-conference proceedings of Operations Research 2013 Conference (preprint: [Heismann and Borndörfer, 2013a]).

Contents

Chapter 1

Terminology and Notation Related to the Hypergraph Assignment Problem (HAP)

The purpose of this chapter is to introduce the hypergraph assignment problem (HAP) and related structures. We assume that the reader is familiar with the basic terminology in graph and hypergraph theory as well as combinatorial optimization. Nevertheless, since some of the concepts are used in slightly different versions in different publications, we will shortly state also this non-HAP-specific terminology and notation that will be used throughout this thesis for disambiguation.

To view the HAP imbedded in the standard framework of combinatorial optimization problems on graphs and hypergraphs, we begin with Section 1.1 on terminology and notation in this area. Then we introduce the HAP and its underlying objects in Section 1.2. Section 1.3 is an overview of the terminology and notation from polytope theory used throughout this thesis. The terminology from complexity theory that will be used in what follows is summarized in Section 1.4. Further, linear and integer programming will play an important role in this thesis. An extensive survey of the theory and methods from this field can be found, for instance, in [Schrijver, 1998].

1.1 Graphs and Hypergraphs

The subject of this thesis is a problem on hypergraphs. Graphs will be widely used to employ results known for well-studied problems such as assignment, matching, or flow. Also, we will sometimes translate structures on hypergraphs

to easier-to-handle graph structures. We will define graphs and several concepts for them as a special case of hypergraphs and use the same notation for both. For a deeper introduction to graph and hypergraph theory, and combinatorial optimization, see, for example, [Grötschel et al., 1988].

Definition 1.1.1. A *hypergraph* $G = (V, E)$ is a pair of a *vertex* set V and a set $E \subseteq 2^V \setminus \{\emptyset\}$ of non-empty subsets of V called *hyperedges*. We denote by $|e|$ the *size* of the hyperedge $e \in E$, and call a hyperedge of size 2 an *edge*. A hyperedge of size greater than 2 is called a *proper* hyperedge. If all hyperedges have size k, i. e., $|e| = k$ for all $e \in E$, G is called *k-uniform*. If all hyperedges are edges, i. e., the hypergraph G is 2-uniform, G is also called a *graph*.

For a vertex subset $W \subseteq V$, we define the *incident hyperedges*

$$\delta_G(W) := \{e \in E : \ e \cap W \neq \emptyset, e \setminus W \neq \emptyset\}$$

to be the set of all hyperedges having at least one vertex in both W and $V \setminus W$. We use the notation $\delta(W)$ instead of $\delta_G(W)$ if the hypergraph is clear from the context. We also write $\delta_G(v) := \delta_G(\{v\})$ if v is a vertex.

For hypergraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, G_1 is called a *subgraph* of G_2 if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

In the hypergraph literature, different types, such as partial hypergraphs or subhypergraphs, of what we call a subgraph are distinguished. Since this more general definition is sufficient for what we do, we use the notion of a subgraph from graph theory as defined above to simplify the terminology.

Definition 1.1.2. Let $G = (V, E)$ be a hypergraph. A sequence

$$(v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n)$$

with $n > 0$, $v_i \in V$ for $i \in \{0, \ldots, n\}$ is called a $[v_0, v_n]$-*path* if for $i, j \in \{1, \ldots, n\}$ the edges $e_i \in E$ fulfill the condition $v_{i-1}, v_i \in e_i$ and the vertices fulfill the condition $v_i \neq v_j$ for $i \neq j$. A $[u, v]$-path with $u = v$ is also called a *cycle*. $G = (V, E)$ is called *connected* if for all $u, v \in G$ with $u \neq v$ there exists a $[u, v]$-path in G. A *connected component* of a hypergraph is a maximal connected subgraph of G with respect to hyperedge inclusion.

We will now state a definition for graphs only. The following notions can also be defined in hypergraphs. However, since this is more complicated and we will need them only for graphs, we here focus exclusively on the graph case.

Definition 1.1.3. A *forest* is a graph without cycles. A forest which is connected is called a *tree*. A *spanning tree* of a graph $G = (V, E)$ is a tree $G' = (V, E')$ which is a subgraph of G having the same vertex set as G.

In the following, we collect some facts on trees and forests that will be important later. Assume that $G = (V, E)$ is a tree. Note that then for $u, v \in V$ there exists a unique $[u, v]$-path in G. Otherwise the concatenation of such a path and another path backwards (with deletion of the last vertex of the first one) would be a cycle, which is a contradiction to G being a tree. Each tree has at least two *leaves*, i. e., vertices that have only one incident edge. If the tree would not have any leaves, starting at some vertex and then always going to a neighbor different than the predecessor (that exists because the vertex has more than one incident edge), which cannot have been visited before (because a tree does not contain a cycle), would lead to an infinite path. If the tree would have only one leaf the same procedure starting at the leaf would find an infinite path. Further, $|E| = |V| - 1$, which can be proven by induction on the cardinality of V. For a forest with k connected components, we can easily conclude that -1 has to be replaced by $-k$ since each connected component of a forest is a tree. Inserting an edge $e = \{u, v\}$ to the tree G changes $|E|$ but not $|V|$ and leads therefore to a cycle in G. This cycle is unique since otherwise G would have had more that one $[u, v]$-path before the insertion of e. This fact is used in the network simplex algorithm, which we will employ in Chapter 7.

Hypergraphs can be represented not only as a pair of sets but also in terms of a matrix, the so-called incidence matrix. This matrix will reappear later as the coefficient matrix in the integer linear programming formulations of several combinatorial optimization problems, especially the HAP.

Definition 1.1.4. Let $G = (V, E)$ be a hypergraph. The matrix

$$A(G) := (a_{ve})_{v \in V, e \in E} \in \mathbb{R}^{V \times E}$$

with

$$a_{ve} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases}$$

is called the *incidence matrix* of G.

For $W \subseteq V$, $F \subseteq E$, we denote by $A_{WF} \in \mathbb{R}^{W \times F}$ the submatrix of A which consists only of the rows for $v \in W$ and the columns for $e \in F$, i. e., $A_{WF} = (a_{ve})_{v \in W, e \in F}$. We also write $A_{W\cdot}$ for A_{WE}, the submatrix of the rows for W, and $A_{\cdot F}$ for A_{VF}, the submatrix of the columns for F. Further, for the rows and columns of A we use the notation $A_{v\cdot} := A_{\{v\}\cdot}$ and $A_{\cdot e} := A_{\cdot\{e\}}$, respectively.

Also, such an incidence matrix can be used to define a hypergraph as follows. A *0/1-matrix* $A \in \{0, 1\}^{V \times E}$ can be interpreted as a hypergraph $G(A) := (V, E)$ with $e \in E$ defined by the column $A_{\cdot e}$ of A. We set

$$e = \{v \in V : \ a_{ve} = 1\}.$$

Note that this definition implies that for a hypergraph G', $G(A(G')) = G'$, and for a 0/1-matrix A', $A(G(A')) = A'$.

The feasible solutions of the hypergraph assignment problem are a special type of partitionings. Some of our results for the HAP hold for partitioning problems in general. Therefore, we now define partitionings and also their relaxations—packings and coverings.

Definition 1.1.5. Let $G = (V, E)$ be a hypergraph. A *packing* $H \subseteq E$ in G is a subset of pairwise disjoint hyperedges, i. e., for all $e_1, e_2 \in H$ with $e_1 \neq e_2$, $e_1 \cap e_2 = \emptyset$. A packing in a graph is also called a *matching*.

A *covering* $H \subseteq E$ in G is a subset of hyperedges that cover all vertices, i. e., $\bigcup H = V$.

A *partitioning* $H \subseteq E$ in G is a subset of hyperedges that covers every vertex exactly once and is therefore both a packing and a covering, i. e., for all $e_1, e_2 \in H$ with $e_1 \neq e_2$, $e_1 \cap e_2 = \emptyset$, and $\bigcup H = V$. A partitioning in a graph is also called a *perfect matching*.

For an example of a hypergraph with a packing, partitioning, and covering, see Figure 1.1.

We are usually interested in a packing, partitioning or covering with minimum cost, which motivates the following definition.

Definition 1.1.6. A *cost function* $c_S : S \to \mathbb{R}$ maps a set S to the reals. For $T \subseteq S$ let

$$c_S(T) := \sum_{s \in T} c_S(s).$$

The optimization problems dealing with packings, partitionings and coverings can be stated as follows.

Problem 1.1.7 (Set Packing (SSP) / Partitioning (SPP) / Covering (SCP) Problem)**.**

Input: A pair (G, c_E) consisting of a hypergraph $G = (V, E)$ and a cost function $c_E : E \to \mathbb{R}$.

Output: A minimum cost packing/partitioning/covering in G w. r. t. c_E, i. e., a packing/partitioning/covering H^* in G such that

$$c_E(H^*) = \min\{c_E(H) : \ H \text{ is a packing/partitioning/covering in } G\},$$

or the information that no packing/partitioning/covering exists.

The set packing, partitioning and covering problem can be described by the following IPs, respectively.

$$\begin{aligned}
\underset{x\in\mathbb{R}^E}{\text{minimize}} \quad & \sum_{e\in E} c_E(e)x_e && && \text{(SSP)}\\
\text{subject to} \quad & \sum_{e\in\delta(v)} x_e \le 1 && \forall v \in V && \text{(i)}\\
& x \ge 0 && && \text{(ii)}\\
& x \in \mathbb{Z}^E && && \text{(iii)}
\end{aligned}$$

$$\begin{aligned}
\underset{x\in\mathbb{R}^E}{\text{minimize}} \quad & \sum_{e\in E} c_E(e)x_e && && \text{(SPP)}\\
\text{subject to} \quad & \sum_{e\in\delta(v)} x_e = 1 && \forall v \in V && \text{(i)}\\
& x \ge 0 && && \text{(ii)}\\
& x \in \mathbb{Z}^E && && \text{(iii)}
\end{aligned}$$

$$\begin{aligned}
\underset{x\in\mathbb{R}^E}{\text{minimize}} \quad & \sum_{e\in E} c_E(e)x_e && && \text{(SCP)}\\
\text{subject to} \quad & \sum_{e\in\delta(v)} x_e \ge 1 && \forall v \in V && \text{(i)}\\
& x \ge 0 && && \text{(ii)}\\
& x \in \mathbb{Z}^E && && \text{(iii)}
\end{aligned}$$

They involve a binary variable x_e for the choice of a hyperedge $e \in E$. Constraints (SSP) (i), (SPP) (i), and (SCP) (i) guarantee that every vertex is covered by at most, exactly, and at least one hyperedge, respectively. (SSP) (ii) and (iii), (SPP) (ii) and (iii), (SCP) (ii) and (iii) are the non-negativity and integrality constraints.

Let

$$\begin{aligned}
P(\text{SSP}) &:= \text{conv}\{x \in \mathbb{R}^E : \text{ (SSP) (i)–(iii)}\},\\
P(\text{SPP}) &:= \text{conv}\{x \in \mathbb{R}^E : \text{ (SPP) (i)–(iii)}\},\\
P(\text{SCP}) &:= \text{conv}\{x \in \mathbb{R}^E : \text{ (SCP) (i)–(iii)}\}
\end{aligned}$$

and

$$P_{\mathrm{LP}}(\mathrm{SSP}) := \{x \in \mathbb{R}^E : \text{ (SSP) (i)–(ii)}\},$$
$$P_{\mathrm{LP}}(\mathrm{SPP}) := \{x \in \mathbb{R}^E : \text{ (SPP) (i)–(ii)}\},$$
$$P_{\mathrm{LP}}(\mathrm{SCP}) := \{x \in \mathbb{R}^E : \text{ (SCP) (i)–(ii)}\}$$

be the polytopes associated with the integer programs (SSP), (SPP), (SCP) and their LP relaxations, respectively.

If two hyperedges in a hypergraph have some vertex in common, they cannot be both part of a packing (and therefore also of a partitioning)—they are "in conflict". All conflicts can be described using a so-called conflict graph.

Definition 1.1.8. Let $G = (V, E)$ be a hypergraph. We call the graph $\mathrm{conf}(G) = (V_{\mathrm{conf}}, E_{\mathrm{conf}})$ with $V_{\mathrm{conf}} = E$ and $E_{\mathrm{conf}} = \{\{e_1, e_2\} \subseteq E : \ e_1 \neq e_2, e_1 \cap e_2 \neq \emptyset\}$ the *conflict graph* of G.

Vertices of the conflict graph that are connected by some edge correspond to hyperedges having a conflict in the original hypergraph. Therefore, a hyperedge set $H \subseteq E$ in a hypergraph $G = (V, E)$ is a packing if and only if the set of vertices $H \subseteq V_{\mathrm{conf}}$ in the conflict graph of G fulfills the following condition: Every pair $\{u, v\}$ of vertices $u, v \in H$ is not an edge contained in E_{conf}. We call such a set of vertices in the conflict graph a stable set.

Definition 1.1.9. A *stable set* $S \subseteq V$ in a graph $G = (V, E)$ is a subset of vertices such that for all $v_1, v_2 \in S$, $\{v_1, v_2\} \notin E$.

For an example of a conflict graph and a stable set, see Figure 1.2. The converse relation between stable sets and packings is also true. If $G = (V, E)$ is a graph, then we can construct a hypergraph G' such that G is its conflict graph and therefore, again, a correspondence between the stable sets and packings exists. We will describe two possibilities to do this below. The set packing problem is therefore sometimes also called the *stable set problem* if viewed in the conflict graph representation. We hence use the abbreviation SSP for the set packing problem to distinguish it from the set partitioning problem (SPP).

The first possibility is to define $G' = (V', E')$ by

$$V' := \{\{v\} : \ v \in V\} \cup E,$$
$$E' := \{b(v) : \ v \in V\},$$
$$b(v) := \{v' \in V' : \ v \in v'\}.$$

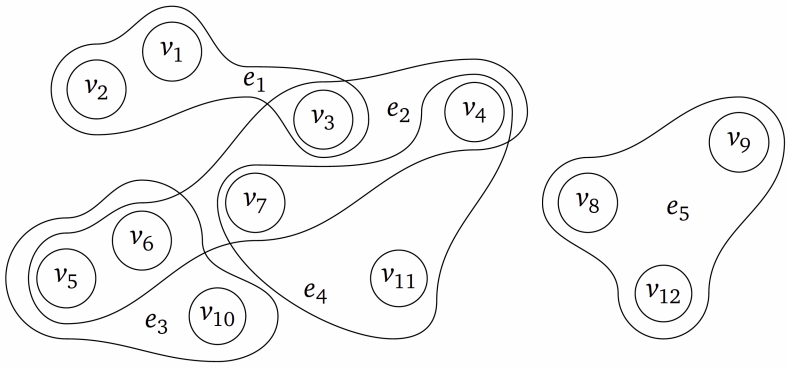


Figure 1.1: Hypergraph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_{12}\}$ and hyperedge set $E = \{e_1, e_2, \ldots, e_5\}$ where $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4, v_5, v_6, v_7\}$, $e_3 = \{v_5, v_6, v_{10}\}$, $e_4 = \{v_4, v_7, v_{11}\}$, $e_5 = \{v_8, v_9, v_{12}\}$. The vertices are drawn as small circles, and hyperedges surround the vertices they contain. $\{e_1, e_3, e_4, e_5\}$ is a partitioning (and therefore also a packing and covering) in G. $\{e_2\}$ is another packing in G. A further example of a covering in G is $\{e_1, e_2, e_3, e_4, e_5\}$.

Then G is the conflict graph of G' since there is an edge $e = \{u, v\} \in E$ if and only if $b(u), b(v) \in E'$ both contain some vertex in V', namely, e. b is a bijection between the stable sets in the original graph G and the packings in the hypergraph G'.

The second construction works as follows. Let $\mathcal{Q}$ be the set of all inclusion-wise maximal subsets Q of V such that for all $v_1, v_2 \in Q$ with $v_1 \neq v_2$, an edge $\{v_1, v_2\} \in E$ exists. These sets are called maximal cliques. Let $b(v) = \{Q \in \mathcal{Q} : v \in Q\}$ for each vertex $v \in V$ of the original graph G. Then G is the conflict graph of $G' = (\mathcal{Q}, \{b(v) : v \in V\})$, and, again, b is a bijection between the stable sets in the original graph G and the packings in the hypergraph G'. This holds because for two vertices $u, v \in V$ there is a edge $e = \{u, v\} \in E$ that connects them if and only if the sets $b(u)$ and $b(v)$ both contain some $Q \in \mathcal{Q}$, namely, some maximal clique Q that contains e.

Such sets of vertices Q in the conflict graph as used in the construction above will also play an important role in Chapter 6. We state them now in terms of the underlying hypergraph.

Definition 1.1.10. A *clique* in (the conflict graph of) a hypergraph $G = (V, E)$ is a set $Q \subseteq E$ of hyperedges such that every two hyperedges $e_1, e_2 \in Q$ have at least one vertex in common, i. e., $e_1 \cap e_2 \neq \emptyset$. A clique Q is a *maximal clique* if there is no clique $Q' \supset Q$ containing Q and in addition other hyperedges. We denote by $\mathcal{Q}$ the set of all maximal cliques.

Associated with the clique Q is the clique inequality $\sum_{e \in Q} x_e \leq 1$.

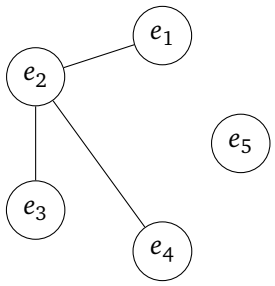


Figure 1.2: Conflict graph $\text{conf}(G) = (V_{\text{conf}}, E_{\text{conf}})$ of the hypergraph G in Figure 1.1 with $V_{\text{conf}} = \{e_1, e_2, \ldots, e_5\}$ and $E_{\text{conf}} = \big\{\{e_1, e_2\}, \{e_2, e_3\}, \{e_2, e_4\}\big\}$. The packings $\{e_1, e_3, e_4, e_5\}$ and $\{e_2\}$ in G are stable sets in $\text{conf}(G)$.

Every feasible solution of (SSP) and therefore also (SPP) fulfills every clique inequality. Clique inequalities for maximal cliques imply all other clique inequalities, which are therefore redundant.

1.2 Bipartite Graphs and Hypergraphs

The combinatorial optimization problems stated in this chapter so far were defined for general graphs or hypergraphs. Set packing, partitioning and covering problems on graphs can be solved efficiently—however, for hypergraphs these problems are $\mathcal{NP}$-hard [Garey and Johnson, 1979]. The HAP is a set partitioning problem on hypergraphs with a special structure, which we call bipartite hypergraphs. Although the HAP is nonetheless $\mathcal{NP}$-hard, it has much more structure than set partitioning problems in general. This can be exploited to generate results which otherwise were not possible. We have already given an impression of this in the preface. In the following chapters, the structure of bipartite hypergraphs will enable us to employ methods that were developed for graphs.

Definition 1.2.1. A hypergraph $G = (U \dot\cup V, E)$ is called *bipartite* if its vertex set can be written as the disjoint union of two vertex sets U and V such that the vertex sets have the same size $|U| = |V|$, and every hyperedge $e \in E$ has the same number $|e \cap U| = |e \cap V| > 0$ of vertices in U and V. We then represent G also as a triple $G = (U, V, E)$.

Bipartite graphs are usually defined without the restriction that the two vertex sets must have the same size. However, if $|U| \neq |V|$ for a bipartite graph or hypergraph $G = (U, V, E)$, the set of feasible solutions for the assignment or

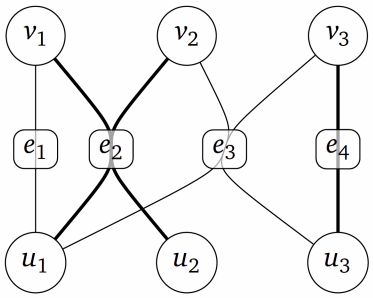


Figure 1.3: Visualization of the bipartite hypergraph $G = (U, V, E)$ with $U = \{u_1, u_2, u_3\}$, $V = \{v_1, v_2, v_3\}$, $E = \{e_1, e_2, e_3, e_4\}$, $e_1 = \{u_1, v_1\}$, $e_2 = \{u_1, u_2, v_1, v_2\}$, $e_3 = \{u_1, u_3, v_2, v_3\}$, $e_4 = \{u_3, v_3\}$. Vertices are circles, edges have square labels, the hyperedges of the hyperassignment $\{e_2, e_4\}$ are drawn with thick lines.

hypergraph assignment problem, respectively, is empty. Since this is the context in which we will use this terminology, the last definition is as stated. Further, bipartite graphs have the property that all of its cycles consist of an even number of edges. This is not true for bipartite hypergraphs.

Note that for a bipartite hypergraph $G = (U, V, E)$ and $W \subseteq U$ or $W \subseteq V$, the Definition 1.1.1 of $e \in \delta(W)$ can be simplified to $e \cap W \neq \emptyset$.

As partitionings in bipartite hypergraphs with equally sized vertex sets are usually given a special name—they are called *assignments*—we now also introduce a special name for partitionings in bipartite hypergraphs.

Definition 1.2.2. Let $G = (U, V, E)$ be a bipartite hypergraph. A partitioning in G is also called a *hyperassignment*.

Figure 1.3 illustrates a hyperassignment in a bipartite hypergraph. We now define the problem that is studied in this thesis.

Problem 1.2.3 (Hypergraph Assignment Problem (HAP))**.**

Input: A pair (G, c_E) consisting of a bipartite hypergraph $G = (U, V, E)$ and a cost function $c_E : E \to \mathbb{R}$.

Output: A minimum cost hyperassignment in G w. r. t. c_E, i. e., a hyperassignment H^* in G such that

$$c_E(H^*) = \min\{c_E(H) : H \text{ is a hyperassignment in } G\},$$

or the information that no hyperassignment exists.

The canonical integer linear program for the HAP is the following. It is the same as for general set partitioning problems. However, we will state it now explicitly since we will do manipulations on this IP which are not possible for general SPPs.

$$\begin{aligned}
\underset{x\in\mathbb{R}^E}{\text{minimize}} \quad & \sum_{e\in E} c_E(e)x_e && && \text{(HAP)}\\
\text{subject to} \quad & \sum_{e\in\delta(v)} x_e = 1 && \forall v \in U\cup V && \text{(i)}\\
& x \geq 0 && && \text{(ii)}\\
& x \in \mathbb{Z}^E && && \text{(iii)}
\end{aligned}$$

Let

$$P(\text{HAP}) := \text{conv}\{x \in \mathbb{R}^E : \text{ (HAP) (i)–(iii)}\}$$

and

$$P_{\text{LP}}(\text{HAP}) := \{x \in \mathbb{R}^E : \text{ (HAP) (i)–(ii)}\}$$

be the polytopes associated with the integer program (HAP) and its LP relaxation, respectively.

Unlike in the graph case, bipartite hypergraphs can have a complex structure, which, of course, cannot be avoided. What we can do, however, is to study a certain "normal form" with a "graph-type appearance" which we find easier to analyze. Our normal form is based on a partitioning of the vertex set that allows to capture the local structure of a hyperassignment in terms of what we call "configurations". We will show in Section 3.2 that every hypergraph can be polynomially transformed into a partitioned hypergraph in such a way that there is a one-to-one correspondence between the hyperassignments in the associated HAP instances.

Definition 1.2.4. A bipartite hypergraph $G = (U, V, E)$ is called *partitioned* with *maximum part size* $d \in \mathbb{N}$ if there exist pairwise disjoint $\leq d$-element sets $U_1, \dots, U_p$ and $V_1, \dots, V_q$ called the *parts* of G such that $\dot{\bigcup}_{i=1}^{p} U_i = U$, $\dot{\bigcup}_{i=1}^{q} V_i = V$, and

$$E \subseteq \bigcup_{i=1}^{p}\bigcup_{j=1}^{q} 2^{U_i\cup V_j},$$

i. e., every hyperedge intersects exactly one part in U and one part in V. In other words, every hyperedge in a partitioned bipartite hypergraph runs from a part of G on the U-side to a part on the V-side. We, in short, call a partitioned bipartite hypergraph a partitioned hypergraph.

For an example of a partitioned hypergraph see Figure 1.4. Note that every hypergraph can be viewed as partitioned if we allow $|U| = |V|$ as the maximum part size. Section 3.2, however, shows a polynomial transformation to a partitioned hypergraph where the maximum part size is equal to half of the maximum hyperedge size. There is a cost-preserving bijection between the hyperassignments in the two hypergraphs.

As special partitioned hypergraphs, we introduce complete partitioned hypergraphs, which will be the subject of, e. g., our analyses of random instances in Chapter 4 and Section 7.1.

Definition 1.2.5. The partitioned hypergraph $G_{k,n} = (U, V, E)$ with n parts $U_1, \dots, U_n$ on the U-side and n parts $V_1, \dots, V_n$ on the V-side, all of size k, and hyperedge set

$$E = \left\{U' \cup V' : \ |U'| = |V'|, U' \subseteq U_i, V' \subseteq V_j \text{ for some } i, j \in \{1, \dots, n\}\right\}$$

is called the *complete partitioned hypergraph* with n k-element parts on the U-side and the V-side.

We now introduce the notion of a configuration to describe the local structure of a hyperassignment H at a part Π, i. e., the possible sets $H \cap \delta(\Pi)$.

Definition 1.2.6. Let $\Pi \in \{U_1, \dots, U_p, V_1, \dots, V_q\}$ be a part of a partitioned hypergraph. We define the set of all *configurations* associated with Π to be

$$\mathscr{C}_\Pi = \left\{C \subseteq \delta(\Pi) : \ \Pi \subseteq \bigcup_{e \in C} e \text{ and } e_1 \cap e_2 = \emptyset \ \forall e_1, e_2 \in C \text{ with } e_1 \neq e_2\right\}.$$

We write $\mathscr{C}_U := \bigcup_{i=1}^{p} \mathscr{C}_{U_i}$, $\mathscr{C}_V := \bigcup_{i=1}^{q} \mathscr{C}_{V_i}$, and $\mathscr{C} := \mathscr{C}_U \cup \mathscr{C}_V$.

A configuration $C \in \mathscr{C}_\Pi$ associated with part Π, w. l. o. g. $\Pi \subseteq U$, is a subset of pairwise disjoint hyperedges that connect all and only the vertices in Π on the U-side with some vertices on the V-side of G, see Figure 1.5 for an illustration. A hyperassignment H induces a configuration $H \cap \delta(\Pi)$ at every part Π.

Another special type of hypergraphs $G = (U \dot\cup V, E)$ that has been investigated in the literature has the property that for each hyperedge $e \in E$,

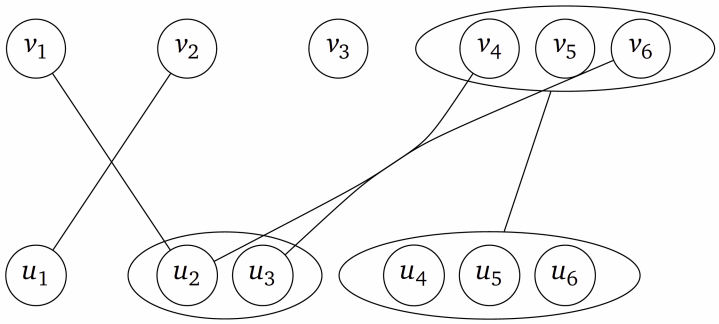


Figure 1.4: Visualization of a partitioned hypergraph with maximum part size $d = 3$, parts $\{u_1\}, \{u_2, u_3\}, \{u_4, u_5, u_6\}$ and $\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5, v_6\}$, and hyperedges $\{u_1, v_2\}, \{u_2, v_1\}, \{u_2, u_3, v_4, v_6\}, \{u_4, u_5, u_6, v_4, v_5, v_6\}$. The vertices of each part with more than one vertex are surrounded by an ellipse in the picture. For partitioned hypergraphs we visualize the hyperedges which connect all the vertices from one part with all the vertices from another part by drawing just a line that connects the two ellipses surrounding the vertices of the part.

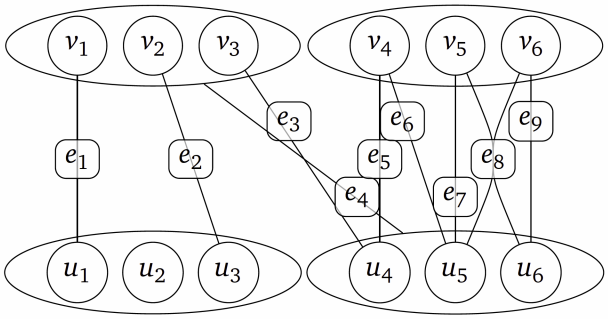


Figure 1.5: In this partitioned hypergraph, the set of all configurations for the part $\Pi = \{u_4, u_5, u_6\}$ is $\mathscr{C}_\Pi = \{\{e_3, e_6, e_9\}, \{e_3, e_7, e_9\}, \{e_3, e_8\}, \{e_5, e_7, e_9\}, \{e_5, e_8\}, \{e_4\}\}$.

$|e \cap U| = 1$ and $|e \cap V| \geq 1$. We can write the HAP in a partitioned hypergraph $G' = (U', V', E')$ with the parts $U'_1, \ldots, U'_n$ on the U'-side and cost function $c_{E'} : E' \to \mathbb{R}$ also as a partitioning problem in this type of hypergraph. This will be done using the configurations $C \in \mathscr{C}_{U'_i}$ for the parts on the U'-side. Let

$$\begin{aligned} U &:= \{U'_1, U'_2, \ldots, U'_n\}, \\ V &:= V', \\ E &:= \left\{ e(C_i) : \; C_i \in \mathscr{C}_{U'_i}, i \in \{1 \ldots n\} \right\} \end{aligned}$$

with

$$\begin{aligned} e(C'_i) &:= \{U'_i\} \cup \bigcup_{e' \in C'_i} \left(V \cap e'\right), \\ c_E(e) &:= \min_{C' \in \mathscr{C}_U :\; e(C') = e} c_{E'}\left(C'\right) \qquad \forall e \in E. \end{aligned}$$

Each hyperedge in E then covers a vertex in U that represents some part U'_i and the vertices in $V = V'$ that some configuration for U'_i covers. In this way, there is a cost-preserving bijection between the hyperassignments in G' and the partitionings in G. It maps the hyperassignments in G to the set of all hyperedges associated with the configurations on the U'-side that are induced by this hyperassignment. We call this representation of the HAP in G the *configurations representation*. It will be of interest regarding the literature discussed in Section 2.3.

1.3 Polytopes

This section summarizes the standard polytope-related terminology and notation. It will be needed especially in Chapter 5, where we deal with a facet classification algorithm but also throughout the thesis in connection with polytopes describing the feasible solutions of combinatorial optimization problems. A deeper introduction to polytope theory can be found, for example, in [Ziegler, 1995].

Definition 1.3.1. A *polyhedron* P is the solution set of a finite system of linear equations and inequalities, in other words, P can be described as

$$P = \{x \in \mathbb{R}^n : \; Ax \leq b, \; Cx = d\}$$

for some $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, C \in \mathbb{R}^{l \times n}, d \in \mathbb{R}^l$. A bounded polyhedron is called a *polytope*. The *dimension* $\dim(P)$ of P is the cardinality of a maximum affinely independent subset minus one.

In the following we will only consider polytopes as feasible solutions of combinatorial optimization problems are usually bounded. Furthermore, we will not give the explicit dimensions of real vectors or matrices whenever they are clear from the context.

An inequality $a^T x \leq b$ is called a *valid inequality for* P if it is satisfied by all elements of P. For a valid inequality, the set $F = \{x : a^T x = b\} \cap P$ is called a *face* of P. A face F is called a *facet* of P if $\dim F = \dim P - 1$. The inequality inducing the facet is called *facet-defining*. A face consisting of a single element is called a *vertex*. We denote by $\text{vert}(P)$ the set of all vertices of P.

A linear representation $\{x : Ax \leq b,\ Cx = d\}$ of a polytope is called an *$\mathscr{H}$-representation of P*.

The *convex hull* $\text{conv}(X)$ of a finite set $X = \{x_1, x_2, \ldots, x_t\} \subseteq \mathbb{R}^n$ is the set of all vectors z which can be written as a *convex combination* $z = \sum_{i=1}^{t} \lambda_i x_i$ with $0 \leq \lambda_i \leq 1$ for $i \in \{1, \ldots, t\}$ and $\sum_{i=1}^{t} \lambda_i = 1$. Analogously, the *linear hull* $\text{span}(X)$ of X, also called the *linear span*, is the set of all vectors z which can be written as a *linear combination* $z = \sum_{i=1}^{t} \lambda_i x_i$, $\lambda_i \in \mathbb{R}$ for $i \in \{1, \ldots, t\}$. If $0 \in \mathbb{R}^n$ can be written as a linear combination of the vectors in X, we say that these vectors are *linearly dependent*. Otherwise they are *linearly independent*. We then also say that one of the vectors in X is linearly independent of the others, i. e., cannot be written as a linear combination of them.

It is a fundamental theorem in polyhedral theory (see, e. g., [Weyl, 1934]) that a polytope P can also be described as the convex hull of its vertices. For $V = \text{vert}(P)$, we call $\text{conv}(V)$ the *$\mathscr{V}$-representation of P*.

1.4 Complexity

We will not cover terminology from complexity theory in detail here. However, we want to informally remind the reader of the complexity classes that appear in this thesis.

$\mathscr{P}$ is the set of all decision problems that can be solved in polynomial time. $\mathscr{NP}$ is the set of all decision problems for which the answer "yes" can be verified using a polynomial size certificate in polynomial time. *$\mathscr{NP}$-complete* is the set of all decision problems such that a polynomial time algorithm for each of them would imply a polynomial time algorithm for all problems in $\mathscr{NP}$. They are "the hardest" problems in $\mathscr{NP}$. The complexity class *$\mathscr{NP}$-hard* is the set of all problems that are "at least as hard" as the problems in the complexity class $\mathscr{NP}$-complete. In particular, it contains the optimization versions of the $\mathscr{NP}$-complete decision problems.

A *polynomial-time approximation scheme (PTAS)* for a minimization or max-

imization problem takes as input a problem instance and some $\epsilon > 0$, and finds a feasible solution with value within the factor of $(1+\epsilon)$ or $(1-\epsilon)$ of the optimal solution value, respectively, in polynomial time. An optimization problem is $\mathscr{APX}$-hard if the existence of a PTAS for it would imply that $\mathscr{P} = \mathscr{NP}$.

A detailed introduction to complexity theory can, for instance, be found in [Garey and Johnson, 1979].

Chapter 2

Literature Overview

The HAP has, to the best of our knowledge, not been studied by other authors in the literature before. However, several related problems on graphs and hypergraphs are subject of previous and current research in graph and hypergraph theory as well as combinatorial optimization. This chapter provides an overview of the problems and results in the field.

Since the HAP is a set partitioning problem on hypergraphs, we have included results on the set partitioning problems in graphs and hypergraphs. These are in Section 2.1 the perfect matching problem for graphs in the general case and the assignment problem for graphs in the the bipartite case. For hypergraphs, we summarize results on the set partitioning problem and its relaxations, the set packing and set covering problem in Section 2.2. Further, problems on specially structured hypergraphs that can be related to the HAP in the configurations representation are part of our overview in Section 2.3. Also, since flow on graphs is closely related to the assignment and the matching problem (they can be written as such [Anstee, 1987]), we have included minimum cost flow on graphs in Section 2.1 as well on as hypergraphs in Section 2.4.

See Figure 2.1 for a structured illustration of the topics included in this literature overview.

2.1 Assignment, Perfect Matching, Flow

The related problems on graphs can be regarded as solved to the greatest possible extent. The polytopes of the LP relaxations for the assignment and minimum cost flow problem, as well as all other problems with a coefficient matrix that is totally unimodular are integral [Schrijver, 2003]. For the perfect matching polytope, the complete description is known [Edmonds, 1965a]. The non-trivial

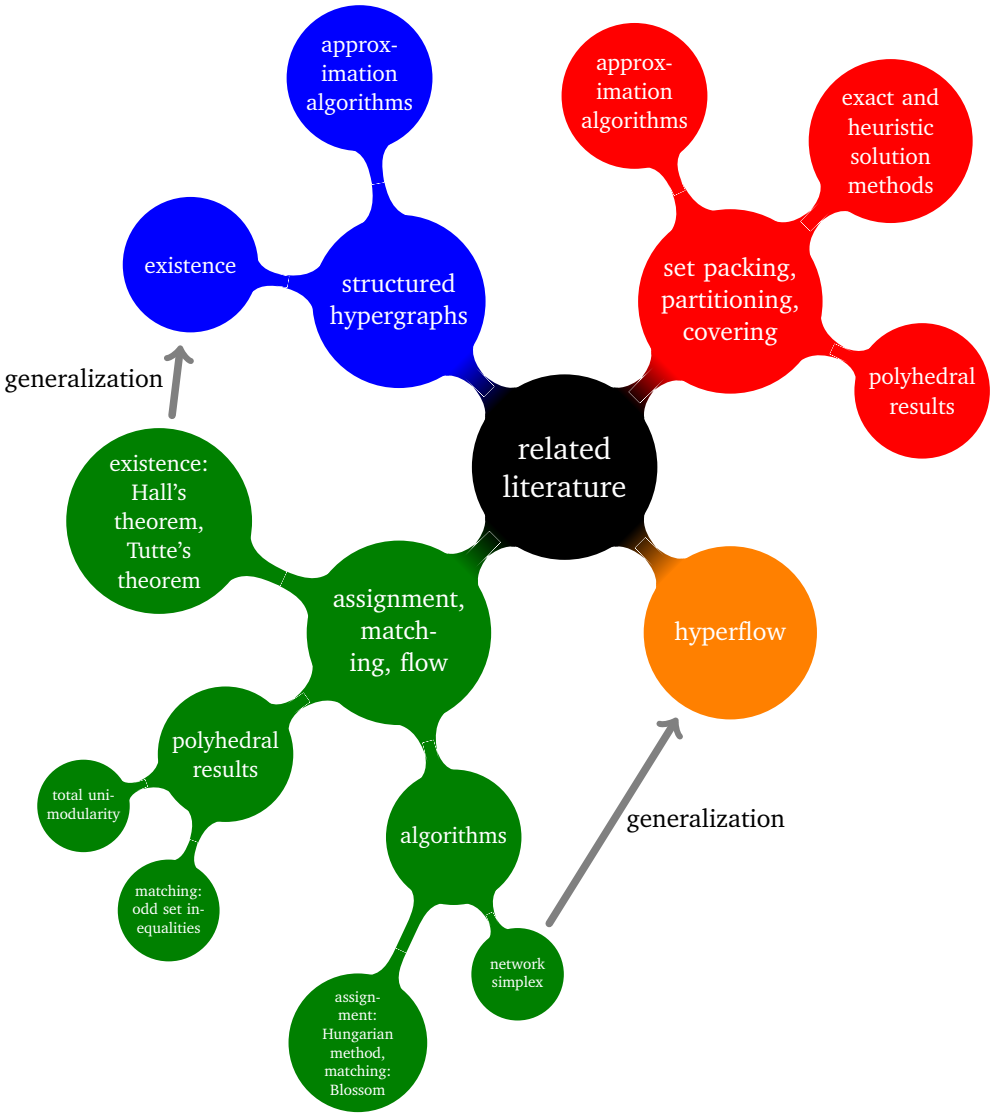


Figure 2.1: Topics included in the literature overview.

facets are called odd set inequalities, and a polynomial time separation algorithm for these has been developed [Padberg and Rao, 1982].

Besides the therefore polynomial algorithms using linear programming, also combinatorial solution methods have been developed. Since the development of the Hungarian algorithm, a primal-dual method, in the 1950s, it is known how to solve the assignment problem in polynomial time [Kuhn, 1955; Munkres, 1957], and other efficient and practically fast algorithms have been developed [Burkard et al., 2012]. In 1965, the first efficient combinatorial algorithm, called the blossom algorithm, for the unweighted version (this means that all costs are equal, the usual optimization problem is then called weighted) of the perfect matching problem was developed [Edmonds, 1965b]. Based on the polyhedral theory from [Edmonds, 1965a], also the weighted version can be solved combinatorially [Lovász and Plummer, 1986]. For the minimum cost flow problem in graphs, several polynomial combinatorial algorithms are known, see, for example, [Ahuja et al., 1993] for an overview. Also, for minimum cost flow the steps of the simplex algorithm can be done using combinatorial operations ("network simplex algorithm") [Orlin, 1997] based on the fact that bases have a spanning tree representation.

For the assignment problem also a necessary and sufficient condition for the existence of a feasible solution, called Hall's theorem [Hall, 1935], is known. It can be generalized to the matching problem and is then called Tutte's theorem [Lovász and Plummer, 1986].

2.2 Set Packing, Covering, and Partitioning

The HAP is a special set partitioning problem and therefore related to the SSP, the SPP, and the SCP. These problems are $\mathcal{NP}$-hard [Garey and Johnson, 1979]. The associated polyhedra $P_{\text{LP}}(\text{SSP})$, $P_{\text{LP}}(\text{SPP})$, $P_{\text{LP}}(\text{SCP})$ are in general not integral. Still, the large research interest in these problems has also led to several positive results on these problems.

Set Packing It was proven that set packing is hard to approximate within $|E|^{1-\epsilon}$ [Håstad, 1996]. The polynomial algorithm with the best approximation factor $\frac{2(\Delta+1)}{3}$ for a maximum hyperedge size Δ independent of $|E|$ that can be found in the literature is achieved by local search [Chandra and Halldórsson, 2001].

$P_{\text{LP}}(\text{SSP})$ is integral for perfect coefficient matrices. Perfect coefficient matrices are exactly those for which the dual of the LP relaxation of (SSP) has an integral optimal solution for integral cost functions, i. e., total dual integrality

holds [Lovász, 1972]. The strong perfect graph conjecture introduced in [Berge, 1961] characterizes hypergraphs with perfect coefficient matrices in terms of odd holes and odd antiholes. Its correctness was shown in [Chudnovsky et al., 2006].

Starting with [Padberg, 1973], several facet classes of the set packing polytope have been found. For some of them polynomial time separation algorithms are known. For an overview, see [Borndörfer, 1998]. Several methods for obtaining facets of $P(\text{SSP})$ can also be found in [Cánovas et al., 2002]. Recently, a survey on the polyhedral results for the set packing (and also covering) problem has been published in [Bentz et al., 2012].

A detailed overview of branch-and-cut methods for the set packing problem as well as a bibliography of other exact solution methods like branch-and-bound or constraint programming can be found in [Rebennack, 2009].

Set Covering The set covering problem is hard to approximate within a factor better than $\ln(\Delta)$ for a maximum hyperedge size Δ [Feige, 1995]. The best currently known polynomial approximation algorithm for the set covering problem with the maximum number k of hyperedges incident to a vertex guarantees an approximation factor of $(k-1)\left(1-e^{-\frac{\ln(\Delta)}{k-1}}\right)+1$, which for very large k tends to $\ln(\Delta)+1$ [Saket and Sviridenko, 2012]. For a recent survey on approximation algorithms for the SCP, see [Shahrokhi, 2009].

The set covering polytope $P(\text{SCP})$ is integral if and only if the coefficient matrix is ideal [Lehman, 1979]. Facets with coefficients in $\{0,1,2\}$ and lifting procedures for them have been characterized in [Balas and Ng, 1989a,b]. The analysis has been extended to facets with coefficients in $\{0,1,2,3\}$ [Saxena, 2004a,b]. The known facets can be used as cutting planes to improve the upper bounds obtained by the LP relaxation, however, this is done only seldom in practice [Caprara et al., 2000]. For a survey on the polyhedral results, see, again, [Bentz et al., 2012].

Exact algorithms for the set covering problem often employ branch-and-bound techniques [Beasley, 1987; Beasley and Jörnsten, 1992]. Heuristic approaches are, e. g., based on Lagrangian relaxation [Beasley, 1990; Balas and Carrera, 1996]. An overview of heuristic and exact algorithms for the set covering problem can be found in [Caprara et al., 2000].

Set Partitioning Since it is $\mathcal{NP}$-hard to even find a feasible solution for the set partitioning problem, polynomial approximation algorithms for the general case cannot be developed.

P_{LP}(SPP) is integral for so-called balanced coefficient matrices [Berge, 1972; Fulkerson et al., 1974]. It was proven in [Berge, 1972] that a matrix is balanced if and only if all of its submatrices are perfect. The same is true if "perfect" is replaced by "ideal".

Further, P_{LP}(SPP) possesses a certain adjacency property: For every two integral vertices x_1, x_2 associated with the bases B_1, B_2 in the simplex algorithm, there exists a sequence of bases $B_1, B_1', B_2', \ldots, B_k', B_2$ that are all associated with integral solutions such that each basis is adjacent to the next one in the sequence [Balas and Padberg, 1972]. However, running the simplex algorithm such that it traverses this sequence of bases might be impossible since it would have to do degenerate steps in directions that are not allowed. To solve this problem, [Balas and Padberg, 1975] propose the composite columns method that combines columns of the simplex tableau to perform allowed steps on only integral solutions and leads to an integral simplex method for the set partitioning problem. To find a composite column, an exponential number of combinations might have to be enumerated. [Rönnberg and Larsson, 2009] show how the composite columns method can be combined with column generation.

All valid inequalities for P(SSP) and P(SCP) are also valid for P(SPP). Further valid—and under certain conditions facet-defining—inequalities based on logical implications of each vertex having exactly one hyperedge that covers it can be found in [Balas, 1977].

Exact algorithms for the SPP use, for instance, branch-and-cut [Hoffman and Padberg, 1993; Borndörfer, 1998] or Lagrangean relaxation [Wedelin, 1995]. Combinations of these methods are also used as heuristics [Atamtürk et al., 1996]. A heuristic method based on linear programming was proposed in [Linderothy et al., 1999].

2.3 Structured Hypergraphs

Despite the great variety of results on set packing, partitioning and covering problems in general, results that can be related to these problems specifically in bipartite hypergraphs are hard to find. A special type of the hypergraph $G = (U \dot\cup V, E)$ with a structure close to the one of bipartite hypergraphs that has been studied in the SSP, SPP, or SCP related literature is as follows. The vertex set of the hypergraph is the union of the disjoint sets U and V, and for each hyperedge $e \in E$, $|e \cap U| = 1$ and $|e \cap V| \geq 1$. As discussed on page 11, the HAP in what we call its configurations representation is formulated in this type of hypergraphs.

We will now discuss results for problems in hypergraphs with such a struc-

ture, and how they can be interpreted within the HAP framework.

Hypermatching Assignment. The hypermatching assignment problem is introduced and studied in [Cygan et al., 2013]. The input of the problem is a hypergraph with two types of vertices. Some vertices represent clients, others represent goods. In the representation from above, the client vertices are in U and the good vertices are in V. Each client has some budget that can be spent. Each hyperedge contains one client and some fixed number k of goods, and a cost as well as a profit value is assigned to it. It represents that the client would buy the set of goods, pay its cost and obtain the profit. The client is only interested in buying all goods from one hyperedge at once, not just a subset of them. A feasible solution is a set of hyperedges such that each good is contained in at most one hyperedge, i. e., bought by at most one client, and the cost sum of the set of hyperedges for each client is at most the client's budget. An optimal solution maximizes the sum of the profits of the clients.

Assume that the costs of all hyperedges as well as all the budgets are equal to some constant number. This implies that each client can buy at most one set of goods. Then, using the configurations representation, we can view the hypermatching assignment problem as the set packing relaxation of the hypergraph assignment problem for partitioned hypergraphs with $|\Pi| = k$ for all parts Π on one side, say U'. However, this is a set packing relaxation with a special constraint: For the parts on the U'-side, either all or none of the vertices are covered by some hyperedge in the packing.

For the hypermatching assignment problem, a randomized $(k + 1 + \epsilon)$-approximation algorithm based on so-called Lasserre hierarchies was developed. This result directly implies an approximation algorithm for the special set packing relaxation of the hypergraph assignment problem described above. The idea of Lasserre hierarchies (see, e. g., [Laurent, 2001] for a survey) is to strengthen the LP relaxation by introducing additional variables describing sets of $0/1$-variables of cardinality at most $t+1$. Such an additional variable is equal to 1 if all the variables in the set have value 1, and 0 otherwise. In each round, t can be increased. If t is large enough, the strengthened LP becomes integral. In [Cygan et al., 2013], Lasserre hierarchies are used for an LP relaxation with variables describing sets of the most profitable hyperedges that a certain client can buy without violating the budget constraints. In our case these would be just single configurations. The projection of the strengthened LP for $t = 1$ to the original variables is used to generate a feasible solution by rounding and a greedy removal of edges to respect the budget constraints.

The other results presented in the cited article [Cygan et al., 2013] on the

hypermatching assignment problem deal with unweighted cases, that is, all hyperedges in the configurations representation would have the same cost.

Combinatorial Auctions. In combinatorial auctions (see [De Vries and Vohra, 2003] for a survey), bidders from the bidder set U submit bids for each subset of items from the item set V. The hyperedges $E = \{\{u\} \cup W : u \in U, W \in 2^V\}$ describe the different bids, and $c_E(\{u\} \cup W) \geq 0$ is the value of the bid of bidder $u \in U$ on the item combination W. The combinatorial auction problem (CAP) asks how the auctioneer should determine which bidder should get which items if the auctioneer has the objective to maximize his profit. This can be viewed as the set packing problem in $G = (U \dot\cup V, E)$ with cost function c_E.

The set packing relaxation of the HAP in the configurations formulation in $G' = (U' \dot\cup V', E')$ with costs $c_{E'} \leq 0$ and the same special constraint as for Hypermatching Assignment (for the parts on the U'-side, either all or none of the vertices are covered by some hyperedge in the packing) can then also be described as a CAP. We set $U := U'$, $V := V'$, E as defined above and

$$c_E(e) = -\min_{e' \in E' : e' \subseteq e} c_{E'}(e').$$

Most methods that have been applied to solve the CAP are those that are used for general set packing problems. Special positive results for the CAP, for example, approximation results [Dobzinski and Schapira, 2006; Feige and Vondrak, 2006], usually employ conditions on the bids, namely, sub- or supermodularity. Submodularity means that for all $u \in U, S, T \subseteq V$,

$$c_E(\{u\} \cup S) + c_E(\{u\} \cup T) \geq c_E(\{u\} \cup S \cup T) + c_E(\{u\} \cup (S \cap T)).$$

For supermodularity, the $\geq$ sign is replaced by a $\leq$ sign.

Sub- or supermodularity does not hold for the combinatorial auction problem constructed above from the set packing relaxation of the HAP in the configurations formulation. If $S, T \subseteq V'$ are non-empty and disjoint, and for some $u \in U$, the hyperedge $\{u\} \cup S \cup T \in E'$, has a non-negative cost, then

$$\begin{aligned} c_E(\{u\} \cup S) + c_E(\{u\} \cup T) &= 0 \\ &< c_E(\{u\} \cup S \cup T) = c_E(\{u\} \cup S \cup T) + c_E(\{u\} \cup (S \cap T)). \end{aligned}$$

On the other hand, if $S, T \subseteq V'$ are disjoint sets such that $\{u\} \cup S, \{u\} \cup T \in E'$ and $c_{E'}(\{u\} \cup S) < c_{E'}(\{u\} \cup T)$, then

$$\begin{aligned} &c_E(\{u\} \cup S) + c_E(\{u\} \cup T) > \\ &\qquad c_E(\{u\} \cup S) = c_E(\{u\} \cup S \cup T) = c_E(\{u\} \cup S \cup T) + c_E(\{u\} \cup (S \cap T)). \end{aligned}$$

Systems of Disjoint Representatives. The problem to decide whether a bipartite hypergraph contains a hyperassignment can be related to the theory of systems of disjoint representatives, see [Aharoni and Haxell, 2000], if using the configurations formulation. A hyperassignment in a partitioned hypergraph selects for each part U_i exactly one configuration $C \in \mathscr{C}_{U_i}$ that covers some vertices $C \cap V$. The vertex sets $C \cap V$ are disjoint (they actually form a partition of V) and therefore can be seen as a system of disjoint representatives in the hypergraph system $\{\{C \cap V : C \in \mathscr{C}_{U_i}\} : i = 1, \ldots, p\}$. Conversely, every system of disjoint representatives in this hypergraph system gives rise to a hyperassignment since the number of covered vertices in V must be equal to $|U| = |V|$.

The existence of a system of disjoint representatives and hence the existence of a hyperassignment can be checked using a generalization of Hall's theorem proposed in [Aharoni and Haxell, 2000] which, however, involves a super-exponential number of conditions. The proof is topological.

Another Hall-type theorem for systems of disjoint representatives was proposed in [Peng and Sissokho, 2013]. It uses an auxiliary graph with two vertex sets. The first vertex set V_1 describes—formulated in the bipartite hypergraph terminology—the parts on the U-side. The second vertex set V_2 describes all the possible sets $C \cap V$ for configurations $C \in \mathscr{C}_U$ for parts in U. For each part U_i, edges from some edge set E_0 connect the vertex in V_1 for the part with the vertices from V_2 describing the sets $|C \cap V|$ for $C \in \mathscr{C}_{U_i}$. Edges in a second edge set R connect pairs of vertices in V_2 if the two corresponding sets for the vertices have a non-empty intersection. A hyperassignment then corresponds to a matching in the graph (V_1, V_2, E_0) that covers all vertices in V_1 but at most one vertex in V_2 from each edge in R, and vice versa. The theorem then just says that a hyperassignment exists if and only if there exists a subset $V_2' \subseteq V_2$ of vertices from V_2 that are a stable set in (V_2, R) such that the bipartite graph $\left(V_1, V_2', \{e \in E_0 : e \subseteq V_1 \cup V_2'\}\right)$ contains an assignment. The existence of an assignment can be checked by Hall's theorem for bipartite graphs. To enumerate all possible choices of V_2', however, stable sets have to be enumerated (which is $\mathscr{NP}$-hard since checking the existence of a stable set is already $\mathscr{NP}$-complete), and their number might be exponential.

2.4 Flow in Directed Hypergraphs

The HAP can be related to the more general minimum cost hyperflow problem with integrality constraints, as we will show now.

The hypergraph assignment problem can be stated as a *minimum cost hyperflow problem* with integrality constraints on a so-called (directed) *B-hypergraph*,

see [Cambini et al., 1992, 1997; Jeroslow et al., 1992]. A B-hypergraph (backward hypergraph) $D = (N, A)$ consists of a vertex set N and a set of *B-hyperarcs* (backward hyperarcs) A. A B-hyperarc $a = (T_a, h_a) \in A$ is pair of a vertex set $T_a \subset N$ (the *tail*) and a vertex $h_a \in N \setminus T_a$ (the *head*); it is supposed to be directed from the tail to the head. "Flow multipliers" can be associated with B-hyperarcs, but we omit them here. We are further given a *demand vector* $b \in \mathbb{R}^N$ and cost function $c_A : A \to \mathbb{R}$ on the B-hyperarcs. A *hyperflow* $f \in \mathbb{R}^A_{\geq 0}$ is a vector, which associates a flow value with each B-hyperarc such that for all vertices $n \in N$ the demand constraint

$$\sum_{a \in A : n = h_a} f_a - \sum_{a \in A : n \in T_a} f_a = b_n$$

is satisfied. Note that the flow at the head of a B-hyperarc is the same as the flow at *each* of the tail vertices. The problem consists of finding a (not necessarily integral) minimum cost hyperflow f^*, i. e.,

$$\sum_{a \in A} c_A(a) f^*_a = \min \left\{ \sum_{a \in A} c_A(a) f_a : \ f \text{ is a hyperflow in } D \right\}.$$

We can state the HAP in (G, c_E) with $G = (U, V, E)$ as a minimum cost hyperflow problem with integrality constraints in the following way. Let $E = E_1 \uplus E_2$ where $E_1 = \{e \in E : \ |e| = 2\}$ is the set of all edges in E and $E_2 = E \setminus E_1$ the set of all proper hyperedges. For $e \in E_1$, let $\{t_e\} = e \cap U$ and $\{h_e\} = e \cap V$; for $e \in E_2$, let $U_e = U \cap e$ and $V_e = V \cap e$. We construct a B-hypergraph $D = (N, A)$ with vertex set $N = U \uplus V \uplus E_2$ and B-hyperarc set $A = A_1 \cup A_2 \cup A'_2$, $A_1 = \{(\{t_e\}, h_e) : e \in E_1\}$, $A_2 = \{(U_e, e) : e \in E_2\}$, $A'_2 = \{(V_e, e) : e \in E_2\}$. In the cost function, we assign $c_A(\{t_e\}, h_e) = c_E(e)$ to the B-hyperarcs in A_1, $c_A(U_e, e) = c_E(e)$ to the B-hyperarcs in A_2, and cost 0 to all B-hyperarcs in A'_2. We define the demand vector b such that

$$b_n = \begin{cases} -1 & \text{if } n \in U \\ 1 & \text{if } n \in E \\ 1 - |\{e \in E_2 : n \in e\}| & \text{if } n \in V. \end{cases}$$

The idea of this construction is that B-hyperarcs $(\{t_e\}, h_e)$ and (U_e, e) have flow value 1 if e is contained in the hyperassignment, while a B-hyperarc (V_e, e) has flow value 1 if e is *not* contained in the hyperassignment; all other flow values are 0. It can be verified that there is a cost-preserving bijection between hyperassignments in G and 0/1 hyperflows in D. Namely, the following 0/1

hyperflow corresponds to a hyperassignment $H \subseteq E$ in G:

$$f_a = \begin{cases} 1 & \text{if } a = (\{t_e\}, h_e) \in A_1,\ e \in H \\ 0 & \text{if } a = (\{t_e\}, h_e) \in A_1,\ e \notin H \\ 1 & \text{if } a = (U_e, e) \in A_2,\ e \in H \\ 0 & \text{if } a = (U_e, e) \in A_2,\ e \notin H \\ 0 & \text{if } a = (V_e, e) \in A_2',\ e \in H \\ 1 & \text{if } a = (V_e, e) \in A_2',\ e \notin H. \end{cases}$$

For an example, see Figure 2.2.

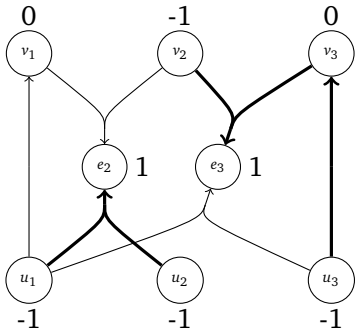


Figure 2.2: The B-hypergraph and the hyperflow (hyperarcs with value 1 are drawn with thick lines) corresponding to the hypergraph and the hyperassignment in Figure 1.3. The numbers next to the vertices are the values of the demand vector.

In contrast to the minimum cost flow problem on graphs, the hyperflow problem does not necessarily have integer solutions for integral inputs.

A hypergraph network simplex algorithm to compute a (not necessarily integer) optimal solution was proposed in [Cambini et al., 1992]. In contrast to the network simplex algorithm on graphs, the proposed method is, however, not purely combinatorial but basically a Schur-complement method, which can be applied in general to sparse linear programs [Gill et al., 1987]. Operations on triangular matrices are described combinatorially in terms of hypergraphs with a tree structure.

Sufficient conditions for ensuring integrality in terms of so-called gain-free Leontief substitution flows have been investigated in [Jeroslow et al., 1992]. Similar results are not known in our setting; in fact, we will show in the next chapter that the HAP is $\mathcal{NP}$-hard even in very simple cases.

Chapter 3

Complexity and Structure

In this chapter, we begin the investigation of the hypergraph assignment problem. To get an idea of what probably is or is not possible, we start in Section 3.1 with a theorem about the complexity of the HAP. It unfortunately shows that even for HAPs with very structured input, a polynomial time algorithm does not exist unless $\mathscr{P} = \mathscr{NP}$. We also deal with the complexity of the set packing and the set covering relaxation of the HAP. In Section 3.2, we state a result which will legitimate us to concentrate our further analysis of the HAP on partitioned hypergraphs as every HAP can be polynomially reduced to a HAP in a partitioned hypergraph. Section 3.3 finally deals with additional restrictions on the HAP input which lead to cases solvable in polynomial time, and a new approximation algorithm for the set covering relaxation. Our contributions on complexity and approximability are summarized in Table 3.1.

Table 3.1: Complexity of the HAP and its packing and covering relaxations.

maximum hyperedge size	**packing relaxation**	**HAP**	**covering relaxation**
4	complexity open	$\mathscr{NP}$-hard, polynomial special cases	complexity open, 2-approximation algorithm
$2d$, $d \geq 3$	$\mathscr{NP}$-hard	$\mathscr{NP}$-hard, polynomial special case	$\mathscr{NP}$-hard, d-approximation algorithm

3.1 $\mathcal{NP}$-Hardness and $\mathcal{APX}$-Hardness

We will now prove that the HAP is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard using a reduction from the 3-dimensional matching problem. These results already hold for bipartite hypergraphs with a very simple structure, namely, for partitioned hypergraphs with part size at most two.

Theorem 3.1.1. *The hypergraph assignment problem is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard, even for partitioned hypergraphs with maximum part size 2.*

Proof. We will use the in its decision version $\mathcal{NP}$-complete and in its optimization version $\mathcal{APX}$-hard 3-dimensional matching problem [Kann, 1991; Garey and Johnson, 1979, page 46]. The input of the 3-dimensional matching problem is a hypergraph $(X \cup Y \cup Z, T)$, $T \subseteq 2^{X \dot\cup Y \dot\cup Z}$ such that $|X| = |Y| = |Z|$ and

$$|t \cap X| = |t \cap Y| = |t \cap Z| = 1 \ \forall t \in T.$$

In its decision version, it asks whether a partitioning in this hypergraph exists, i. e., a set $F \subseteq T$ such that each element from $X \dot\cup Y \dot\cup Z$ is contained in exactly one set in F. In the optimization version of the 3-dimensional matching problem, some cost function $c_T : T \to \mathbb{R}$ is given and a partitioning with minimum cost w. r. t. c_T has to be found. Let

$$\begin{aligned} X &= \{x_1, \ldots, x_n\}, \\ Y &= \{y_1, \ldots, y_n\}, \\ Z &= \{z_1, \ldots, z_n\}, \\ T &= \{t_1, \ldots, t_m\} \end{aligned}$$

with $t_r = \{x_{i_r}, y_{j_r}, z_{k_r}\}$, $r = 1, \ldots, m$.

To prove the theorem we first construct an instance of the hypergraph assignment problem having a size that is polynomial in the size of the given 3-dimensional matching problem. We will show that there exists a hyperassignment in the HAP if and only if there exists a partitioning in the 3-dimensional matching problem. This proves that the decision problem version of the HAP in partitioned hypergraphs with maximum part size 2, i. e., the question whether a hyperassignment in a given hypergraph of such a type exists, is $\mathcal{NP}$-complete, and therefore the $\mathcal{NP}$-hardness of the optimization version of the HAP. To prove the $\mathcal{APX}$-hardness, we then show that there exists a cost function for the HAP such that for each hyperassignment in the HAP there exists also a partitioning in the 3-dimensional matching problem that has the same cost.

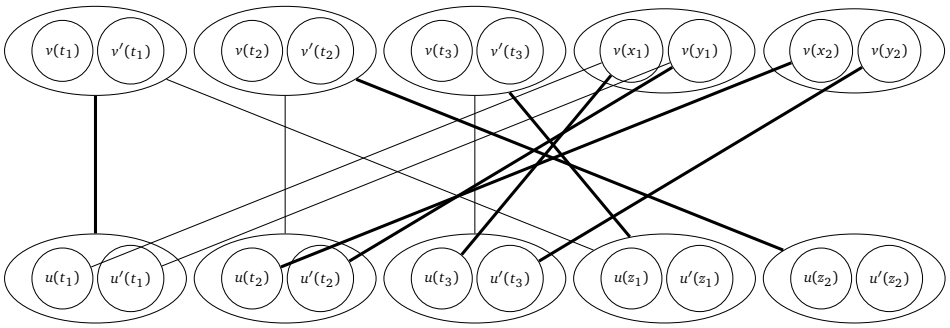


Figure 3.1: Example of the construction of G in the $\mathcal{NP}$-hardness proof for $(X \cup Y \cup Z, T)$ with $n = 2$ and $T = \{t_1, t_2, t_3\}$, $t_1 = \{x_1, y_1, z_1\}$, $t_2 = \{x_2, y_1, z_2\}$, $t_3 = \{x_1, y_2, z_1\}$. The partitioning $F = \{\{x_2, y_1, z_2\}, \{x_1, y_2, z_1\}\}$ gives rise to a hyperassignment in G drawn with thick lines.

Let $G = (U, V, E)$ be a partitioned hypergraph with parts

$$U(z_1) = \{u(z_1), u'(z_1)\},\ \ldots,\ U(z_n) = \{u(z_n), u'(z_n)\},$$
$$U(t_1) = \{u(t_1), u'(t_1)\},\ \ldots,\ U(t_m) = \{u(t_m), u'(t_m)\}$$

on the U-side and

$$V(x_1, y_1) = \{v(x_1), v(y_1)\},\ \ldots,\ V(x_n, y_n) = \{v(x_n), v(y_n)\},$$
$$V(t_1) = \{v(t_1), v'(t_1)\},\ \ldots,\ V(t_m) = \{v(t_m), v'(t_m)\}$$

on the V-side. Let

$$\begin{aligned} E = \big\{\{u(t_r), v(x_{i_r})\} : r \in \{1, \ldots, m\}\big\} \\ \cup \big\{\{u'(t_r), v(y_{j_r})\} : r \in \{1, \ldots, m\}\big\} \\ \cup \big\{\{u(z_{k_r}), u'(z_{k_r}), v(t_r), v'(t_r)\} : r \in \{1, \ldots, m\}\big\} \\ \cup \big\{\{u(t_r), u'(t_r), v(t_r), v'(t_r)\} : r \in \{1, \ldots, m\}\big\}. \end{aligned}$$

For an example of this construction, see Figure 3.1.

Let $H \subseteq E$ be a hyperassignment in G. Each of the vertices $v(x_1), \ldots, v(x_n)$, $v(y_1), \ldots, v(y_n)$, $u(z_1), \ldots, u(z_n)$, $u'(z_1), \ldots, u'(z_n)$ is contained in exactly one hyperedge in H. All such hyperedges contain at least one vertex from one of the parts $U(t_r)$ or $V(t_r)$ for some r. The four vertices $u(t_r)$, $u'(t_r)$, $v(t_r)$, $v'(t_r)$ from the parts $U(t_r)$, $V(t_r)$, $r = 1, \ldots, m$, are contained either in the hyperedge $\{u(t_r), u'(t_r), v(t_r), v'(t_r)\}$ in E (case one) or in the three hyperedges

$\{u(t_r), v(x_{i_r})\}$, $\{u'(t_r), v(y_{j_r})\}$, and $\{u(z_{k_r}), u'(z_{k_r}), v(t_r), v'(t_r)\}$ (case two) in E. Thus, the set of all $\{x_{i_r}, y_{j_r}, z_{k_r}\} \in T$ for which r is associated with case two form a partitioning in $(X \cup Y \cup Z, T)$.

On the other hand, given a partitioning in $(X \cup Y \cup Z, T)$ we get a hyperassignment H in G by choosing the hyperedges associated with case two exactly for those r for which $\{x_{i_r}, y_{j_r}, z_{k_r}\}$ is in the partitioning and the hyperedge associated with case one otherwise.

Choose the cost function $c_E : E \to \mathbb{R}$ defined by

$$c_E(e) := \begin{cases} c_T(t_r) & \text{if } e = \{u(t_r), v(x_{i_r})\} \text{ for some } r \in \{1, \dots, m\} \\ 0 & \text{otherwise} \end{cases}$$

for the HAP. Then the mapping between hyperassignments H in G and partitionings F in $(X \cup Y \cup Z, T)$ described above preserves the costs, i. e., $c_E(H) = c_T(F)$. □

Further results related to the complexity of the HAP, such as the proof of an arbitrarily large LP-IP gap for partitioned hypergraphs with maximum part size two, arbitrarily large determinants of basis matrices, as well as other $\mathcal{NP}$-hardness proofs for less strict cases than in Theorem 3.1.1 can be found in [Heismann, 2010].

We also want to mention here that the set packing and the set covering relaxation of the HAP in bipartite hypergraphs with hyperedge size at least 6 are $\mathcal{NP}$-hard. This can be shown by a reduction from the SSP and the SCP in general hypergraphs. For some set packing or set covering problem instance (G, c_E) with $G = (V, E)$ we will construct a set packing or set covering instance $(G', c_{E'})$, respectively, on the bipartite hypergraph $G' = (U', V', E')$. Let $U' := \{u'(v) : v \in V\}$ and $V' := \{v'(v) : v \in V\}$ be copies of V, and let $E' = \{e'(e) : e \in E\}$ where $e'(e) = \{u'(v), v'(v) : v \in e\}$. Note that the hyperedge $e'(e)$ has the double size of the corresponding hyperedge e. Define $c_{E'}(e'(e)) = c_E(e)$. Then, $b : E \to E', e \mapsto e'$ is a cost-preserving bijection that maps packings resp. coverings in G to packings resp. coverings in G' with the same cost. Since the SSP and SCP are $\mathcal{NP}$-hard for maximum hyperedge size ≥ 3, the set packing and set covering relaxation of the HAP are therefore $\mathcal{NP}$-hard for maximum hyperedge size ≥ 6. We could not find an answer to the complexity question for the set packing and set covering relaxation of the HAP for maximum hyperedge size 4.

3.2 Structural Results

Several results in this thesis are described for the hypergraph assignment problems on partitioned hypergraphs. We will show now that this is not a real restriction, because every bipartite hypergraph G with maximum hyperedge size $2d$ can be polynomially transformed into a partitioned hypergraph G' with maximum part size d in such a way that there exists a cost preserving bijection between the hyperassignments in G and G'.

The idea of the construction is to set up a hypergraph that consists of disjoint copies of the original hyperedges plus some "garbage collection" edges that will match superfluous vertices in a hyperassignment.

Theorem 3.2.1. *Let $G = (U, V, E)$ be a bipartite hypergraph with maximum hyperedge size $2d$ and c_E a cost function. Then there exists a partitioned hypergraph $G' = (U', V', E')$ with maximum part size d and a cost function $c_{E'}$ such that there is a hyperassignment H in G of cost c if and only if there is a hyperassignment H' in G' of the same cost. G' can be constructed in polynomial time.*

Proof. Let

$$U' := \left\{(u,e) : u \in U,\ e \in E,\ u \in e\right\} \cup \left\{u'_i(v) : v \in V,\ i \in \{1,\dots,|\delta(v)|-1\}\right\}$$

and

$$V' := \left\{(v,e) : v \in V,\ e \in E,\ v \in e\right\} \cup \left\{v'_i(u) : u \in U,\ i \in \{1,\dots,|\delta(u)|-1\}\right\}.$$

(u,e), $u'_i(v)$, (v,e), $v'_i(u)$ are new vertices indexed by u and e, i and v, v and e, i and u, respectively. For every $u \in U$ and $v \in V$, order the vertices in $\left\{(u,e) \in U'\right\}$ as $\{u''_1,\dots,u''_{|\delta(u)|}\}$ and those in $\left\{(v,e) \in V'\right\}$ as $\{v''_1,\dots,v''_{|\delta(v)|}\}$, respectively.

For each hyperedge $e \in E$ we construct a "copy" $e' = \{(u,e),(v,e) : u \in U \cap e, v \in V \cap e\} \in E'$ with cost $c_{E'}(e') = c_E(e)$. Further, we construct edge sets of cost zero to control that exactly one of the copies (u,e) or (v,e) of vertex u or v, respectively, is covered by a hyperedge copy in every hyperassignment in G'. We have

$$\begin{aligned} E' := \left\{e' : e \in E\right\} \cup \\ \left\{\{u''_i, v'_j(u)\} :\ (i-j) \in \{0,1\},\ u \in U,\ i \in \{1,\dots,|\delta(u)|\}\right\} \cup \\ \left\{\{u'_j(v), v''_i\} :\ (i-j) \in \{0,1\},\ v \in V,\ i \in \{1,\dots,|\delta(v)|\}\right\} \end{aligned}$$

and set $G' := (U', V', E')$. This construction can be done in polynomial time.

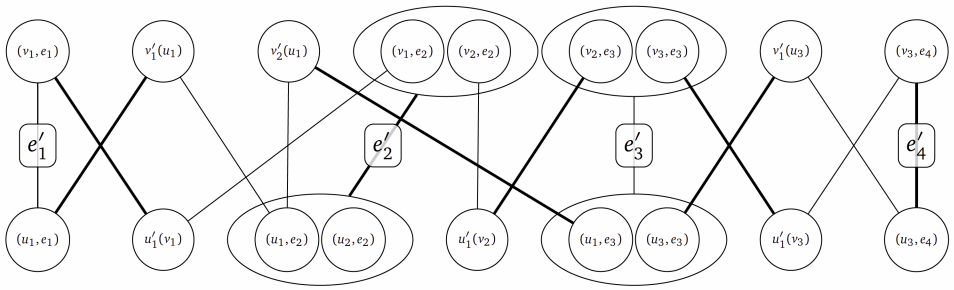


Figure 3.2: The corresponding partitioned hypergraph for the bipartite hypergraph from Figure 1.3. The thick hyperedges show a hyperassignment in both. In this construction the vertices of the type (u,e) or (v,e) for every vertex u or v of the original hypergraph are ordered by the index of the hyperedges.

First of all, we show that G' can be partitioned. By construction, G' is bipartite (and well-defined). Choosing the sets $\{(u,e) : u \in e\}$, $\{(v,e) : v \in e\}$ for every $e \in E$ and the remaining vertices individually as parts produces a partitioned hypergraph. As $|\{U \cap e\}| = |\{V \cap e\}| \leq d$ for each hyperedge e and since every vertex in G' has at most one incident hyperedge that is proper, the maximum part size is d.

Let H be a hyperassignment in G. For every vertex $u \in U$ let $i(u)$ be the index of $u''_{i(u)} = (u,e)$ where e is the unique hyperedge in H containing u, and define $i(v)$ for $v \in V$ analogously; note that $i(u)$ and $i(v)$ depend on H. Let

$$H' := \{e' : e \in H\} \cup \{u''_i, v'_i(u) : u \in U,\ i < i(u)\} \cup \{u''_i, v'_{i-1}(u) : u \in U,\ i > i(u)\} \cup \{u'_i(v), v''_i : v \in V,\ i < i(v)\} \cup \{u'_{i-1}(v), v''_i : v \in V,\ i > i(v)\}.$$

H' contains the copies of the hyperedges in H, and matches the unused vertex copies to the garbage collection vertices $v'_i(u)$ and $u'_i(v)$, see Fig. 3.2 for an illustration. H' is a hyperassignment in G' with cost $c_{E'}(H') = c_E(H)$.

On the other hand, given a hyperassignment H' in G' we can construct a hyperassignment H in G using exactly the hyperedges from E corresponding to the hyperedges $e' \in H'$. Again, $c_{E'}(H') = c_E(H)$. To prove that H is a hyperassignment we need to show that for every vertex $v \in U \uplus V$ of G, H' has only one vertex (v,e) which is contained in a hyperedge e' in H'. For this purpose observe that G' contains $|\delta(v)| - 1$ vertices $u'_i(v)$ and $|\delta(v)|$ vertices (v,e) for every vertex v of G. The garbage collection vertices $u'_i(v)$ have to be matched with vertices (v,e) via garbage collection edges such that exactly one vertex copy (v,e) must be covered by a hyperedge. □

3.3 Polynomially Solvable Cases and Approximation

As we have seen in Section 3.1, the HAP cannot be solved in polynomial time even for partitioned hypergraphs with maximum part size two. Therefore, a subset of HAP instances that might be solved in polynomial time would have to fulfill restrictions on the allowed hyperedges in the hypergraph or on the cost function.

A polynomially solvable type of HAP instances with a restriction of the allowed hyperedges is the HAP in partitioned hypergraphs with a bounded maximum part size d and a bounded number k_U, k_V of parts of size ≥ 2 on the U-side, V-side, respectively. The hyperedges that are not incident to both such a part in U and such a part in V are edges. Therefore, there are at most $k_U k_V (2^d - d - 1)^2$ proper hyperedges in the hyperedge set. Such a HAP can then be solved in polynomial time by branching on the values of the proper hyperedges. Given fixed values of these hyperedges, the remaining hyperedges are all edges and the HAP reduces to an assignment problem. A more sophisticated method to solve such HAPs in polynomial time would be to branch on vertex groupings, see Section 7.1.

As a polynomially solvable class of the HAP for partitioned hypergraphs with maximum part size two and a restricted cost function, we propose a class of cost functions with arbitrary edge costs but proper hyperedge costs following a certain structure. It can be solved combinatorially using a reduction to a minimum cost perfect matching problem, i. e., a set partitioning problem on a graph.

Theorem 3.3.1. *Let $G = (U, V, E)$ be a partitioned hypergraph with maximum part size two and $c_E : E \to \mathbb{R}$ be a cost function. Let $U_1, \dots, U_p$ and $V_1, \dots, V_q$ be the parts of size two on the U- and V-side, respectively. If*

$$E_2 := \left\{ U_i \cup V_j : \ i \in \{1, \dots, p\}, j \in \{1, \dots, q\} \right\} \subseteq E,$$

and there exists a cost function $c : \{U_1, \dots, U_p, V_1, \dots, V_q\} \to \mathbb{R}$ such that

$$c_E(U_i \cup V_j) = c(U_i) + c(V_j) \text{ for all } i \in \{1, \dots, p\}, j \in \{1, \dots, q\},$$

then the HAP with input (G, c_E) can be solved in polynomial time. The costs of edges can be chosen arbitrarily.

Proof. We prove the theorem by reducing the HAP with the given restrictions to a minimum cost perfect matching problem. Minimum cost perfect matching problems can be solved combinatorially in polynomial time [Lovász and Plummer, 1986, p. 370]. The input of the problem will be the graph $G' = (U \dot\cup V, E')$

with the same vertex set as G and some cost function $c_{E'}$. We will partition the set of hyperassignments in G into equivalence classes such that all equivalent hyperassignments have the same cost w. r. t. c_E. Further, we will show that there exists a bijection b between these sets $\{H_1, \ldots, H_k\}$ of equivalent hyperassignments $H_1, \ldots, H_k$ in G and the set of perfect matchings M in G' such that $c_{E'}(M) = c_E(H_1) = \ldots = c_E(H_k)$. Given a perfect matching M in G', a hyperassignment from $b^{-1}(M)$ can be found in polynomial time.

We now describe the edge set E' of the graph $G' = (U \dot\cup V, E')$, and the cost function $c_{E'}$. Let $E_1 := E \setminus E_2$. Note that by construction of G, E_1 is the set of all edges and E_2 the set of all proper hyperedges of G. Let

$$E' := E_1 \cup \{U_i : \ i \in \{1, \ldots, p\}\} \cup \left\{V_j : \ j \in \{1, \ldots, q\}\right\}.$$

Assign the following costs to the edges in E':

$$\begin{aligned} c_{E'}(e) &:= c_E(e) && \text{for } e \in E, \\ c_{E'}(U_i) &:= c(U_i) && \text{for } i \in \{1, \ldots, p\}, \\ c_{E'}(V_i) &:= c(V_i) && \text{for } j \in \{1, \ldots, q\}. \end{aligned}$$

Now, let two hyperassignments H and H' in G be equivalent if and only if $H \cap E_1 = H' \cap E_1$ and therefore $\bigcup(H \cap E_2) = \bigcup(H' \cap E_2)$ holds for the two sets of vertices $\bigcup(H \cap E_2)$ and $\bigcup(H' \cap E_2)$ covered by a proper hyperedge in H and H' respectively. This inherits the property of being an equivalence relation from "=". Since G is partitioned, the sets $\bigcup(H \cap E_2)$ and $\bigcup(H' \cap E_2)$ can also be written as $\bigcup_{i \in I} U_i \cup \bigcup_{j \in J} V_j$ for some index sets $I \subseteq \{1, \ldots, p\}$ and $J \subseteq \{1, \ldots, q\}$. H and H' have the same cost

$$\begin{aligned} c_E(H) = \sum_{e \in H \cap E_1} c_E(e) + \sum_{i \in I} c(U_i) + \sum_{j \in J} c(V_i) \\ = \sum_{e \in H' \cap E_1} c_E(e) + \sum_{i \in I} c(U_i) + \sum_{j \in J} c(V_i) = c_E(H'). \end{aligned}$$

Define b by assigning the perfect matching $M := (E_1 \cap H) \cup \{U_i : i \in I\} \cup \{V_j : j \in J\}$ to these hyperassignments. See Figure 3.3 for an example of the construction.

M is indeed a perfect matching: By the definition of a hyperassignment, each vertex $v \in V$ either belongs to the set $\bigcup(H \cap E_1)$ or to the set $\bigcup(H \cap E_2)$. In the first case, v was covered by exactly one edge $e \in E_1$ in H and is therefore also covered only by e in M. In the second case, v is contained in some part Π of size two in G and covered only by the edge Π in M.

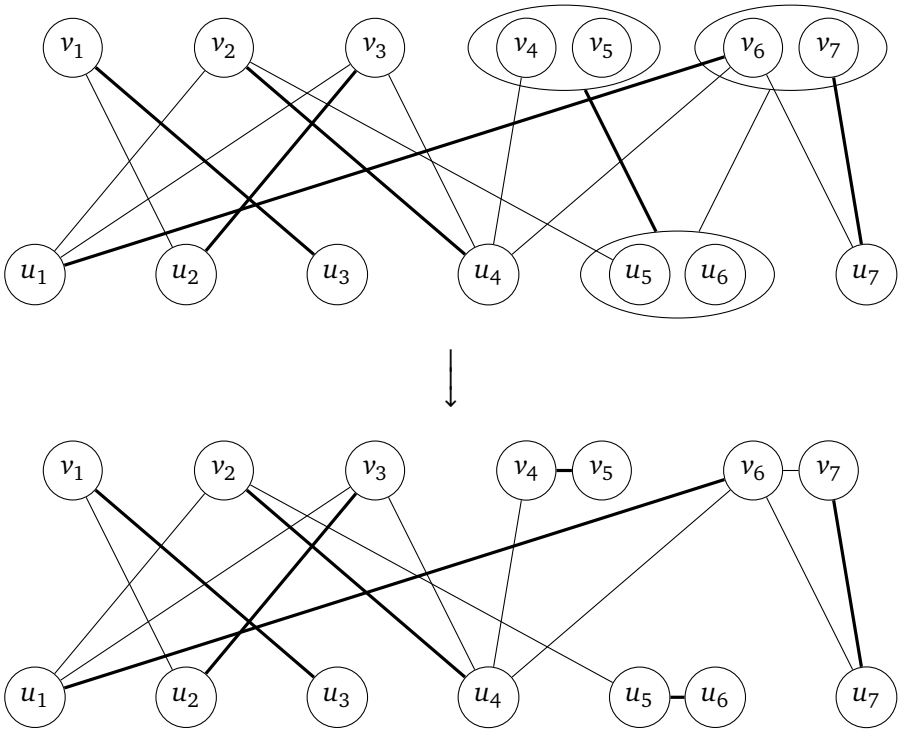


Figure 3.3: Example of the construction from the proof of Theorem 3.3.1. The upper image shows a bipartite hypergraph G with maximum part size two that fulfills the requirements of the theorem. A hyperassignment H is drawn with thick lines. The lower image shows the corresponding graph G' with the perfect matching $b(H)$ drawn with thick lines.

Also,

$$c_{E'}(M) = \sum_{e \in H \cap E_1} c_E(e) + \sum_{i \in I} c(U_i) + \sum_{j \in J} c(V_i) = c_E(H).$$

It remains to show that b is a bijection, i. e., injective and surjective. Injectiveness holds since by construction of M every two hyperassignments whose equivalence classes are mapped to the same perfect matching belong to the same class. To see that b is surjective, first observe that the sets $I := \{i : U_i \in \mathrm{M}\}$ and $J := \{j : U_j \in \mathrm{M}\}$ have the same cardinality because $|U| = |V|$ and the edges in M which are not counted for I and J are all in E_1, and cover therefore the same number of vertices in U and V. We can now construct a hyperassignment H from the equivalence class of $b^{-1}(M)$ as follows. Let w. l. o. g. $I = J = \{1, \ldots, k\}$ (otherwise rename the parts). Then $H = (M \cap E_1) \cup \{U_i \cup V_i : i \in \{1, \ldots, k\}\}$ is such a hyperassignment. □

We also want to show here that the set covering relaxation of the HAP in bipartite hypergraphs with hyperedge size at most $2d$ is d-approximable with a polynomial algorithm (if we assume that all hyperedge costs are positive as this is usually done for the SCP). For $d = 2$ and a large number of hyperedges incident to each vertex, this is better than one can achieve by applying the best-known approximation algorithms for general SCPs to bipartite hypergraphs: The best currently known polynomial approximation algorithm for the set covering problem is the classic greedy method. It guarantees the approximation factor $\sum_{i=1}^{\Delta} \frac{1}{i}$ [Chvatal, 1979] where Δ is the maximum size of a hyperedge. For $\Delta = 2d = 4$, this implies an approximation factor of about 2.083, which is more than $d = 2$.

The approximation factor d for the set covering relaxation of the HAP with maximum hyperedge size $2d$ can be achieved by Algorithm 3.3.1. It transforms the set covering problem in a bipartite hypergraph to a set covering problem in a bipartite graph such that an optimal solutions for the latter implies an approximation for the first. For a bipartite hypergraph $G = (U, V, E)$ with maximum hyperedge size $2d$ and cost function $c_E : E \to \mathbb{R}_{>0}$, the algorithm constructs a bipartite graph $G' = (U, V, E')$ where E' is the set of all edges connecting some vertex from U with some vertex from V that are contained in some hyperedge in E. To get the value of the cost function $c_{E'} : E' \to \mathbb{R}_{>0}$ for the edge $e' \in E'$, we find the hyperedge $e_{e'}$ in E such that its cost divided by $\frac{|e_{e'}|}{2}$ is minimal, and use this value as a $c_{E'}(e')$. Finally, we calculate a minimum cost covering H' w. r. t. $c_{E'}$ in G' and return the set H of all hyperedges $e_{e'} \in E$ for which e' is in H'.

We now shortly explain why Algorithm 3.3.1 is correct. Since the hyperedge $e_{e'}$ is a superset of the hyperedge e', and H' covers U and V (in G'), H is also

Algorithm 3.3.1: d-approximation algorithm for the set covering relaxation of the HAP with maximum hyperedge size $2d$.

Data: bipartite hypergraph $G = (U, V, E)$ with maximum hyperedge size $2d$ and cost function $c_E : E \to \mathbb{R}_{>0}$
Result: covering in G which has cost at most d times the cost of an optimum solution

$E' \leftarrow \emptyset$
foreach $e \in E$ **do** // for all hyperedges in E
  $E'(e) \leftarrow \emptyset$
  foreach $u \in e \cap U$ **do**
    foreach $v \in e \cap V$ **do**
      $E'(e) \leftarrow E'(e) \cup \{u, v\}$ // add to $E'(e)$ all possible pairs of a vertex from U and a vertex from V contained in e
  $E' \leftarrow E' \cup E'(e)$ // add $E'(e)$ to E'
foreach $e' \in E'$ **do**
  $e_{e'} \leftarrow \arg\min_{e \in E:\ e' \subseteq e} \frac{c_E(e)}{0.5 \cdot |e|}$
  $c_{E'}(e') \leftarrow \frac{c_E(e_{e'})}{0.5 \cdot |e_{e'}|}$
$G' \leftarrow (U, V, E')$
$H' \leftarrow$ minimum cost set covering in G' with cost function $c_{E'}$
$H \leftarrow \emptyset$
foreach $e' \in H'$ **do** // for all hyperedges e' in H'
  $H \leftarrow H \cup e_{e'}$ // add to H the hyperedge $e_{e'} \supseteq e'$
return H

a covering (in G). For each hyperedge $e \in H$ there is at least one edge $e' \in H'$ such that $e = e_{e'}$, and $c_{E'}(e') = \frac{c_E(e)}{0.5 \cdot |e|} \geq \frac{c_E(e)}{d}$. Hence, $c_E(H) \leq d \cdot c_{E'}(H')$. On the other hand, if H is a covering in G, then replacing all hyperedges by a set of pairwise disjoint edges whose union is the hyperedge leads to a covering H' in G' with $c_{E'}(H') \leq c_E(H)$. Thus, if H^* is an optimal solution for the set covering problem in G, then Algorithm 3.3.1 calculates a covering H' in G' and returns a covering H for G such that

$$c_E(H) \leq d \cdot c_{E'}(H') \leq d \cdot c_E(H^*).$$

Therefore, the algorithm is a d-approximation algorithm for the set covering problem in bipartite hypergraphs with maximum hyperedge size $2d$. The algorithm has a polynomial running time since the SCP in a graph, also called the minimum edge cover problem, can be solved in polynomial time [Garey and Johnson, 1979], and all other operations can obviously be done in polynomial time, too.

Chapter 4

Optimal Solutions of Random Instances with Standard IP Methods

A way to gain a better understanding of the structure of a combinatorial optimization problem is to analyze random instances, especially their optimal values. For the assignment problem, such results were proposed by computational experiments and then proven theoretically. For a survey on the so-called "random assignment problem" and several of its generalizations, see [Krokhmal and Pardalos, 2009].

In particular, a result for the assignment problem on random instances in complete bipartite graphs and i. i. d. uniform random variables on $[0, 1]$ or i. i. d. exponential random variables with mean 1 as edge costs can be proven. The expected optimal value converges to $\frac{\pi^2}{6} = 1.6449\ldots$ if the number of vertices tends to infinity. This is known as the Conjectures of Mézard and Parisi [Mézard and Parisi, 1985]. The limit is equal for both distributions since it can be proven that only the density at 0 is relevant, and it is equal for both distributions [Aldous, 1992].

In this chapter, we consider a random version of the hypergraph assignment problem in complete partitioned hypergraphs $G_{2,n}$ with all parts having size two. For this hypergraph type, the hyperedge set consists only of edges, and proper hyperedges of size 4. Further, the hypergraph underlies a structure that makes it easy to view it as a combination of two assignment problems, one consisting only of edges, the other of proper hyperedges viewed as edges. However, the coupling of the two assignment problems is such that it involves a choice over an exponential number of possibilities.

For i. i. d. uniform random variables on $[0,1]$ or i. i. d. exponential random variables with mean 1 as hyperedge costs, we first show computational results in Section 4.1 that suggest that the minimum cost of hyperassignments converges to some value around 1.05 with a small standard deviation for large n. Our results also suggest that the optimal value most probably is attained with half of the maximum possible number of proper hyperedges.

Then we prove a lower bound 0.3718 and an upper bound 1.8310 for the expected value of a minimum cost hyperassignment in $G_{2,2n}$ which uses exactly n proper hyperedges for vertex number tending to infinity for the exponential distribution. This will be achieved using a combinatorial argument for the understanding of the bounds computed afterwards and employing results for the random assignment problem in the computation.

In hypergraph assignment problems that arise from practical applications, proper hyperedges are likely to have costs depending on the costs of the edges that are subsets of them. Hyperedges "reward" the choice of a combination of the edges for which the hyperedge is their union. This means that these edges are similar in some way and it is desirable to have a solution with much similarity [Borndörfer et al., 2011]. Such cost functions will be the subject of Section 4.2. We will present computational results and discuss observations regarding the dependence of the number of proper hyperedges in an optimal solution and the optimal value on the number of vertices in the hypergraph and a certain penalty parameter.

4.1 Same Cost Distribution for Edges and Proper Hyperedges

4.1.1 Computational Experiments

See Table 4.1 for computational results for the random hypergraph assignment problem in the bipartite graphs $G_{2,n}$ with i. i. d. uniform random variables on $[0,1]$ or i. i. d. exponential random variables with mean 1 as hyperedge costs. For every n, we give the mean value and standard deviation for 1000 computations (performed by CPLEX 12.5 with the canonical IP formulation (HAP)).

The computational results propose that the expected optimal value converges to a value around 1.05 for both distribution. Although for larger n more hyperedges are contained in a hyperassignment, the optimal value does not increase much. This can be intuitively explained by the fact that for larger n there are also more possible hyperassignments to select from, and the chances to find a hyperassignment that has a low cost are therefore still good even if it will

contain more hyperedges.

Since the standard deviation is relatively small for large n, it suggests that the optimal value for such random HAP instances can be predicted with a high reliability.

Furthermore, in our computational results the optimal value is attained where about half of the maximum possible number of proper hyperedges is used. This proposes that such random HAPs are hard to solve even if one could assume that the optimal hyperassignment is such that about half of the parts on the U-side and on the V-side are incident to proper hyperedges: The number of choices of n parts on the U-side and n parts on the V-side in $G_{2,2n}$ is $\binom{2n}{n}^2$. In the next subsection, we will compute bounds between which the expected optimal value of the minimum cost hyperassignment with such a restriction on the number of proper hyperedges in it lies.

4.1.2 Bounds for the Case of an Exponential Distribution

For partitioned hypergraphs of the type $G_{2,n}$ with parts $U_1, \dots, U_n$ on the U-side and $V_1, \dots, V_n$ on the V-side, the hypergraph assignment problem can be seen as a combination of two assignment problems as follows. Observe that for every hyperassignment H and each $i \in \{1, \dots, n\}$, $\delta(U_i) \cap H$ and $\delta(V_i) \cap H$ consist either of one proper hyperedge or of two edges. If we fix for every part whether it has to be incident to one proper hyperedge or two edges in the hyperassignment, we can restrict the hyperedge set of $G_{2,n}$ to

- the set of edges connecting pairs of vertices from the parts in U and in V that will be incident to edges—this is the first assignment problem, and
- the proper hyperedges $\{U_i \cup V_j\}$ for parts U_i and V_j that are fixed as incident to proper hyperedges—viewing U_i and V_j as single vertices and the hyperedges as edges connecting them, this is the second assignment problem.

The solution of the two assignment problems separately with costs as in the hypergraph assignment problem implies the minimum cost hyperassignment in which the incidence to one proper hyperedge or two edges for each part U_i and V_j is as it was fixed to be.

Thus, the HAP in $G_{2,n}$ can be viewed as a decision for parts that have to be incident to proper hyperedges (same number of parts from U and V, equal to the number of proper hyperedges in a feasible hyperassignment; the other parts will be incident to edges) and then solving the two assignment problems stated above.

Table 4.1: Computational results with random hypergraph assignment problems in $G_{2,n}$ for i. i. d. uniform random variables on $[0,1]$ or i. i. d. exponential random variables with mean 1 as hyperedge costs. The mean optimal values (column 2 and 6) and their standard deviations (column 3 and 7) are rounded to the third decimal place. The number of proper hyperedges in the found optimal hyperassignment (column 4 and 8) and their standard deviations (column 5 and 9) are rounded to one decimal place. 1000 computations were done for each value of n and each distribution.

	uniform on $[0,1]$				exponential with mean 1			
n	o. v.	s. d.	# p. h.	s. d.	o. v.	s. d.	# p. h.	s. d.
10	0.943	0.177	5.5	2.0	1.019	0.206	5.3	2.0
20	1.006	0.136	10.4	2.8	1.039	0.141	10.4	2.8
30	1.018	0.109	15.5	3.4	1.049	0.117	15.3	3.4
40	1.037	0.096	20.7	4.0	1.045	0.097	20.5	3.9
50	1.036	0.085	25.8	4.4	1.054	0.085	25.4	4.3
60	1.044	0.078	31.0	4.8	1.050	0.080	30.6	4.7
70	1.041	0.074	35.8	4.9	1.053	0.079	35.6	5.1
80	1.044	0.070	40.9	5.4	1.054	0.069	40.6	5.4
90	1.044	0.066	45.9	5.8	1.053	0.066	45.9	5.8
100	1.047	0.061	50.9	6.3	1.057	0.063	50.6	6.3
110	1.047	0.058	56.3	6.3	1.054	0.060	56.1	6.4
120	1.048	0.057	61.1	6.6	1.052	0.056	61.1	6.7
130	1.051	0.055	66.4	7.1	1.054	0.053	66.3	6.9
140	1.053	0.054	71.6	7.4	1.053	0.051	71.3	7.1
150	1.051	0.053	76.0	7.7	1.051	0.050	76.2	7.5
160	1.048	0.049	81.6	7.4	1.054	0.048	81.2	7.6

To compute a lower and upper bound on the expected value of a minimum cost hyperassignment in $G_{2,2n}$ with n proper hyperedges, we will use the following result: For a complete bipartite graph with vertex sets of size m and n (here it is assumed that the two vertex sets can have different sizes) and with i. i. d. exponential random variables with mean 1 as edge costs the expected minimum value of the sum of k pairwise disjoint edges (this is called a partial assignment) is

$$E(m,n,k) := \sum_{\substack{i,j\geq 0 \\ i+j\leq k-1}} \frac{1}{(n-i)(m-j)}.$$

This was conjectured in [Coppersmith and Sorkin, 1999] and the first proof appeared in [Linusson and Wästlund, 2004]. There is also shown that for $m = n = k$ this term can be written as

$$E(n,n,n) = \sum_{i=1}^{n} \frac{1}{i^2}.$$

This expected value of the minimum assignment in a complete bipartite graph with two vertex sets of size n with i. i. d. exponential random variables with mean 1 as edge costs is known as Parisi's Conjecture.

Theorem 4.1.1. *For the expected value* $\mathbf{E}$ *of the minimum cost of a hyperassignment in* $G_{2,2n} = (U,V,E)$ *with exactly* n *proper hyperedges and cost function* c_E *with i. i. d. exponential random variables with mean 1 as* $c_E(e)$ *for all* $e \in E$*, for* $n \to \infty$ *the following holds:*

$$0.3718 < \mathbf{E} < 1.8310.$$

Proof. By definition,

$$E(n) := E(2n,2n,n) = \sum_{\substack{i,j\geq 0 \\ i+j\leq n-1}} \frac{1}{(2n-i)(2n-j)}.$$

Using $E(n)$, we can bound the expected value of a hyperassignment in $G_{2,2n}$ with i. i. d. exponential random variables mean 1 as hyperedge costs restricted to the hyperassignments with n proper hyperedges as follows.

For the lower bound, observe that in the best possible hyperassignment the selected n proper hyperedges can be only as good as the n pairwise disjoint proper hyperedges with the least possible cost sum in $G_{2,2n}$. Also, the selected $2n$ edges can be only as good as the $2n$ pairwise disjoint edges with the least possible cost sum in $G_{2,2n}$. Thus, $E(n) + E(2n)$ is a lower bound.

On the other hand, choosing the n pairwise disjoint proper hyperedges with the least possible cost sum in $G_{2,2n}$ and finding the best possible edges for the remaining "unused" vertices leads to an upper bound of $E(n) + E(2n,2n,2n)$.

To transform the two-indexed sum describing $E(n)$ to a sum with only one index, we calculate the difference $D(n) := E(n+1) - E(n)$ and use the recursive formula

$$E(n) = E(1) + \sum_{i=1}^{n-1} D(i) = \frac{1}{4} + \sum_{i=1}^{n-1} D(i). \tag{4.1}$$

We get

$$D(n) = E(n+1) - E(n)$$

$$= E(2n+2, 2n+2, n+1) - E(2n, 2n, n)$$
$$= \sum_{\substack{i,j\geq 0 \\ i+j\leq n}} \frac{1}{(2n+2-i)(2n+2-j)} - \sum_{\substack{i,j\geq 0 \\ i+j\leq n-1}} \frac{1}{(2n-i)(2n-j)}.$$

Shifting the index of the first sum to get the same summand type in both sums yields

$$= \sum_{\substack{i,j\geq -2 \\ i+j\leq n-4}} \frac{1}{(2n-i)(2n-j)} - \sum_{\substack{i,j\geq 0 \\ i+j\leq n-1}} \frac{1}{(2n-i)(2n-j)}.$$

We now split the sums to sums with index range $i, j \geq 0$, $i + j \leq n-4$ so that they can cancel. The remainder is as follows. For the first sum, it is used that it is symmetric in i and j. The term $\frac{(4n+3)^2}{4(n+1)^2(2n+1)^2}$ is the sum of the values where $-2 \leq i, j \leq -1$. This has to be subtracted from the first term as otherwise these values would be counted twice.

$$\begin{aligned} D(n) = 2 \cdot & \sum_{\substack{-2\leq i\leq -1, j\geq -2 \\ i+j\leq n-4}} \frac{1}{(2n-i)(2n-j)} \\ & - \frac{(4n+3)^2}{4(n+1)^2(2n+1)^2} - \sum_{\substack{i,j\geq 0 \\ i+j=n-1}} \frac{1}{(2n-i)(2n-j)} \\ & - \sum_{\substack{i,j\geq 0 \\ i+j=n-2}} \frac{1}{(2n-i)(2n-j)} - \sum_{\substack{i,j\geq 0 \\ i+j=n-3}} \frac{1}{(2n-i)(2n-j)}. \end{aligned}$$

Splitting the first sum into two parts with $i = -1$ and $i = -2$ and substituting j by $a - i$ where $i + j = a$ yields

$$\begin{aligned} D(n) = & \sum_{j=-2}^{n-3} \frac{2}{(2n+1)(2n-j)} + \sum_{j=-2}^{n-2} \frac{2}{(2n+2)(2n-j)} \\ & - \frac{(4n+3)^2}{4(n+1)^2(2n+1)^2} - \sum_{i=0}^{n-1} \frac{1}{(2n-i)(n+1+i)} \\ & - \sum_{i=0}^{n-2} \frac{1}{(2n-i)(n+2+i)} - \sum_{i=0}^{n-3} \frac{1}{(2n-i)(n+3+i)}. \end{aligned}$$

Using the notation $H_n = \sum_{i=1}^{n} \frac{1}{i}$ for the n-th harmonic number and partial fraction decomposition to get denominators linear in n for the last two summations, we get

$$\begin{aligned}
D(n) = &\frac{2H_{2n+2} - 2H_{n+2}}{2n+1} + \frac{2H_{2n+2} - 2H_{n+1}}{2n+2} \\
&- \frac{(4n+3)^2}{4(n+1)^2(2n+1)^2} - \frac{2H_{2n} - 2H_n}{3n+1} \\
&- \frac{2H_{2n} - 2H_{n+1}}{3n+2} - \frac{2H_{2n} - 2H_{n+2}}{3n+3} \\
= &\frac{2H_{2n} + \frac{2}{2n+1} + \frac{2}{2n+2} - 2H_n - \frac{2}{n+1} - \frac{2}{n+2}}{2n+1} \\
&+ \frac{2H_{2n} + \frac{2}{2n+1} + \frac{2}{2n+2} - 2H_n - \frac{2}{n+1}}{2n+2} \\
&- \frac{(4n+3)^2}{4(n+1)^2(2n+1)^2} - \frac{2H_{2n} - 2H_n}{3n+1} \\
&- \frac{2H_{2n} - 2H_n - \frac{2}{n+1}}{3n+2} - \frac{2H_{2n} - 2H_n - \frac{2}{n+1} - \frac{2}{n+2}}{3n+3}.
\end{aligned}$$

Finally, simplification yields

$$\begin{aligned}
D(n) = &-(H_{2n} - H_n)\frac{9n^2 + 11n + 4}{3(n+1)(2n+1)(3n+1)(3n+2)} \\
&+ \frac{8n^2 + 13n + 6}{12(n+1)^2(2n+1)^2(3n+2)}.
\end{aligned}$$

To get bounds on $E(n)$ using Equation (4.1), we first use that

$$\begin{aligned}
&\sum_{n=1}^{\infty} \frac{8n^2 + 13n + 6}{12(n+1)^2(2n+1)^2(3n+2)} \\
&\qquad = -\frac{1}{4} - \frac{\pi}{\sqrt{3}} + \frac{\pi^2}{9} - \frac{10\ln(2)}{3} + \ln(27). \quad (4.2)
\end{aligned}$$

Then, observe that $H_{2n} - H_n$ is a non-negative number monotonically increasing with n. Also, this is an alternating harmonic number that for $n \to \infty$ converges to $\ln(2)$. For $n = 80$, $H_{2n} - H_n$ is equal to

$$\frac{8119740843426279518461644384261262504562959619404143919063830759076 9}{1176719554879018748378908156413626819469883030031412208979707195680 00},$$

which is > 0.69. Therefore, for $n \geq 80$,

$$0.69 < H_{2n} - H_n < \ln(2) \tag{4.3}$$

Now, computing the partial sum

$$\sum_{n=1}^{79} -(H_{2n} - H_n) \frac{9n^2 + 11n + 4}{3(n+1)(2n+1)(3n+1)(3n+2)}$$

exactly and the limes

$$\sum_{n=80}^{\infty} -(H_{2n} - H_n) \frac{9n^2 + 11n + 4}{3(n+1)(2n+1)(3n+1)(3n+2)}$$

after substituting for $H_{2n} - H_n$ the lower and upper bounds given by (4.3), Equations (4.1) and (4.2) yield

$$0.1859 < \lim_{n\to\infty} E(n) < 0.1860.$$

Thus, we get for the lower bound

$$\begin{aligned} \lim_{n\to\infty} (E(n) + E(2n)) &= 2 \cdot \lim_{n\to\infty} E(n) \\ &> 2 \cdot 0.1859 \\ &= 0.3718 \end{aligned}$$

and for the upper bound

$$\begin{aligned} \lim_{n\to\infty} (E(n) + E(2n, 2n, 2n)) &= \lim_{n\to\infty} E(n) + \lim_{n\to\infty} E(2n, 2n, 2n) \\ &< 0.1860 + \frac{\pi^2}{6} \\ &< 1.8310 \end{aligned}$$

□

The upper bound computed in Theorem 4.1.1 is greater than the expected optimal value of the random assignment problem $\frac{\pi^2}{6} = 1.6449\ldots$ if the number of vertices tends to infinity. However, since moving from an assignment problem in a complete bipartite graph with $4n$ vertices on each side, $G_{1,4n}$, to the HAP

in $G_{2,2n}$ just adds more possibilities (still all assignments are feasible solutions but using hyperassignments with proper hyperedges gives additional ones), it is clear that if we do not prescribe the number of proper hyperedges in an optimal solution, the expected optimal value of a hyperassignment in $G_{2,2n}$ will tend to some number $\leq \frac{\pi^2}{6}$. As already discussed, the computational results shown in Table 4.1 suggest that the number will be some value around 1.05, much smaller than $\frac{\pi^2}{6}$.

4.2 Regularity Rewarding Costs

So far, we have dealt in this chapter with random HAPs for which all hyperedge costs are random. However, in instances that are models for practical applications, the proper hyperedge costs will depend on the costs of the edges that are subsets of them. Proper hyperedges model a "reward" for choosing a combination of the edges for which the hyperedge is their union. Ones wants to reward certain edge combination choices since these choices imply a so-called regularity of the solution [Borndörfer et al., 2011]. In such instances, the bipartite hypergraph is partitioned and all the edges that are incident to a certain part in U and a certain part in V are those that are desirable to choose at once. Therefore, the proper hyperedge that is a union of such pairwise disjoint edges has a cost that is less than the sum of the edge costs. If there are different edge combinations that lead to the same hyperedge, the cost is inferred from the edge set with the minimum cost sum.

We now state this more formally.

Definition 4.2.1. Let $G = (U, V, E)$ be a partitioned hypergraph and let $E_1 \subseteq E$ be the set of all its edges. For $e \in E$, let

$$\mathscr{E}(e) := \{E' \subseteq E_1 : \ e_1 \cap e_2 = \emptyset \ \forall e_1, e_2 \in E' \text{ with } e_1 \neq e_2, \bigcup E' = e\}$$

be the set of all pairwise disjoint edge sets with union e. Assume that $|\mathscr{E}(e)| > 0$ for all $e \in E$ (this is a requirement for proper hyperedges—for edges $e \in E_1$, this is true anyway since $\mathscr{E}(e) = \{\{e\}\}$).

For some *penalty* $p \geq 0$, we call a cost function $c_E : E \to \mathbb{R}$ *regularity rewarding* if for all proper hyperedges $e \in E \setminus E_1$,

$$c_E(e) = \min_{E' \in \mathscr{E}(e)} \left(\sum_{e' \in E'} c_E(e') - p \cdot |E'| \right).$$

The greater p the more irregularity is punished and reguarity rewarded. In the vehicle rotation planning model the costs of a hyperedge are determined by

different parameters and if a hyperedge is inclusionwise not maximal, a penalty for irreguarity is added on top [Borndörfer et al., 2011]. Therefore, we call p a penalty instead of a bonus or a reward.

For regularity rewarding cost functions, we have analyzed random HAP instances in $G_{2,n}$ as in Section 4.1. The costs were determined such that for each edge e some random basic cost r_e from a uniform distribution on $[0,1]$ was chosen and then

$$c_E^p(e) := \begin{cases} r_e + p & \text{if } e \text{ is an edge.} \\ \min_{E' \in \mathcal{E}(e)} \sum_{e' \in E'} r_{e'} & \text{if } e \text{ is a proper hyperedge} \end{cases}$$

was defined. c_E^p obviously fulfills the definition of being regularity rewarding.

Figure 4.1 shows the results on the mean optimal value for instances with such a cost function in $G_{2,n}$ for different n and different values of p, for 100 instances for each combination of n and p, and the standard deviations. In Figure 4.2, the mean number of proper hyperedges in the found optimal solution is depicted in the upper plot. The lower plot of this figure shows this number relative to n. For the more precise numbers, see Appendix A. All calculations were done with CPLEX 12.5 with the canonical IP formulation (HAP). We will now discuss observations in these two figures.

Dependence of the optimal solution value and the proper hyperedge number on the penalty and the hypergraph size. The optimal solution value is non-descreasing with an increasing penalty and hypergraph size. We can see that the mean number of proper hyperedges in the optimal solution increases with an increasing value of the penalty p for a fixed n. The larger n the higher the mean relative number of proper hyperedges in the optimal solution for a fixed p. For "large" values of p, the hypergraph assignment problem becomes almost an assignment problem since we can predict with a very high probability that only proper hyperedges will be used. What "large" means depends on n.

Standard deviations. As this was the case for the other two cost functions in Section 4.1, also for the regularity rewarding cost functions the relative standard deviation decreases with an increasing hypergraph size. Therefore, the optimal value of a random instance with a large n can be predicted to lie with a high probability inside of a small range if the mean value for the n and p of the instance is known. Also, the relative standard deviations of the number of proper hyperedges in the optimal solution decreases with an increasing n. This might be connected to what we have discussed previously on the values: If for a fixed p the optimal solutions start to consist almost only of proper hyperedges

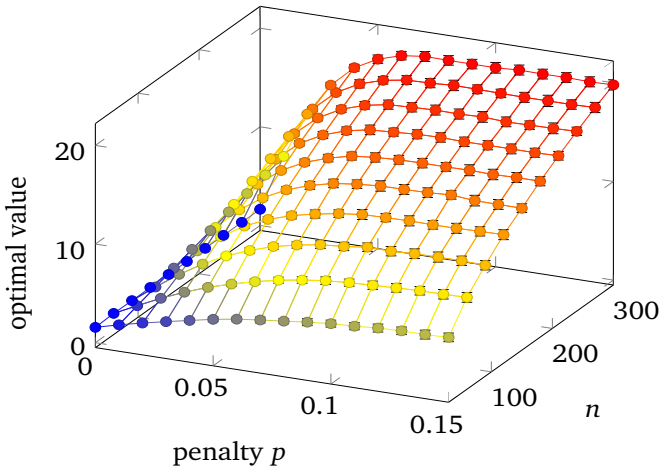


Figure 4.1: Mean optimum value of 100 samples of the HAP in $G_{2,n}$ with a cost function c_E^p for different penalties $p \in \{0.0, 0.01, \ldots, 0.15\}$ and different sizes $n \in \{30, 60, \ldots, 300\}$. The error bars show the standard deviation. They might be not visible if the standard deviation is very small. Tables with the numbers depicted in this figure can be found in Appendix A.

for large n, the standard deviation of the number of proper hyperedges has to be very small for large n.

Mean optimal value of solutions with only proper hyperedges. If the optimal solution consists only of proper hyperedges, then its mean value is not $\leq 2 \cdot \frac{\pi^2}{6}$ as one might expect but much higher. The reason is that the distribution of the sum of two independent uniform random variables in $[0,1]$ is not uniform in $[0,2]$. As we will show now, the probability of proper hyperedges having values close to 0 is much smaller than if their values would be given by a uniform random distribution in $[0,2]$.

For every proper hyperedge e there exist four distinct edges e_1, e_2, e_3, e_4 such that the only possibilities to find pairwise disjoint edges the union of which is e are $e_1 \cup e_2$ and $e_3 \cup e_4$. The cost of e in a regularity rewarding cost function c_E^p is $\min(c_E^p(e_1) + c_E^p(e_2) - 2p, c_E^p(e_3) + c_E^p(e_4) - 2p)$. Thus, to understand the distribution of the cost function for proper hyperedges, we are interested in the distribution of $\min\big(X_1 + X_2, X_3 + X_4\big)$ for the i. i. d. uniform random variables on $[0,1]$ X_1, X_2, X_3, X_4.

The sum of two i. i. d. uniform random variables on $[0,1]$ is the Irwin-Hall

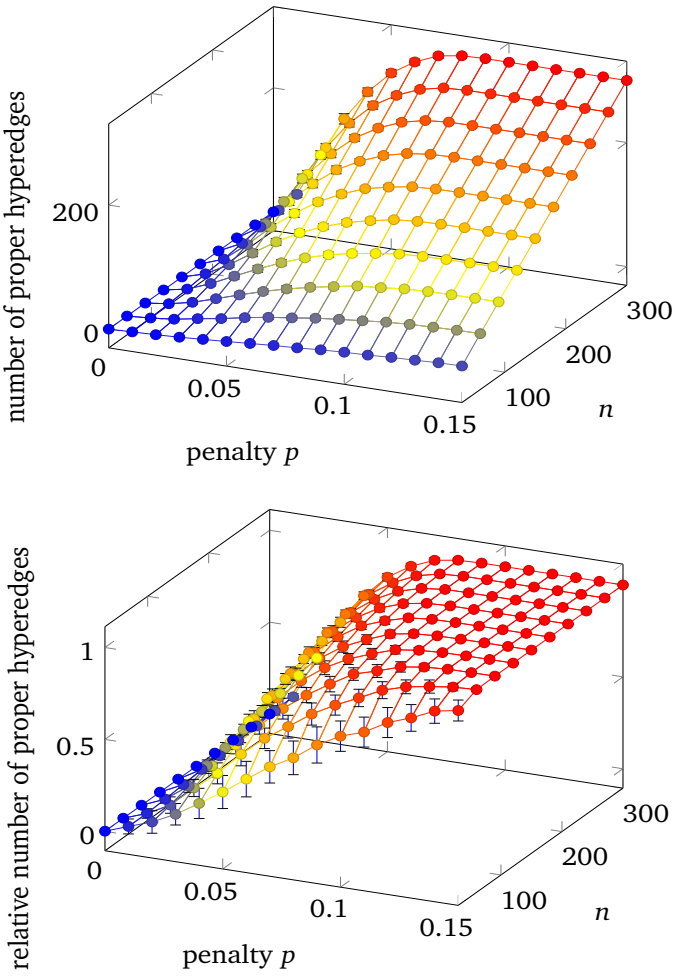


Figure 4.2: Mean absolute (upper figure) and relative (lower figure) proper hyperedge number, i. e., the proper hyperedge number and proper hyperedge number divided by n, in the optimal solution found of 100 samples of the HAP in $G_{2,n}$ with a cost function c_E^p for different penalties $p \in \{0.0, 0.01, \ldots, 0.15\}$ and different sizes $n \in \{30, 60, \ldots, 300\}$. The error bars show the standard deviation. They might be not visible if the standard deviation is very small. Tables with the numbers depicted in this figure can be found in Appendix A.

distribution [Johnson et al., 1995] with the probability density function

$$f_{X_1+X_2}(x) = f_{X_3+X_4}(x) = \begin{cases} x, & \text{if } x \in [0,1] \\ 2-x, & \text{if } x \in [1,2] \\ 0, & \text{otherwise.} \end{cases}$$

This gives the cumulative distribution function

$$F_{X_1+X_2}(x) = F_{X_3+X_4}(x) = \begin{cases} \frac{1}{2}x^2, & \text{if } x \in [0,1] \\ -\frac{1}{2}x^2 + 2x - 1, & \text{if } x \in [1,2] \\ 0, & \text{otherwise.} \end{cases}$$

Thus,

$$\begin{aligned} F_{\min(X_1+X_2, X_3+X_4)}(x) &= 1-(1-F_{X_1+X_2}(x))^2 \\ &= \begin{cases} -\frac{1}{4}x^4 + x^2, & \text{if } x \in [0,1] \\ -\frac{1}{4}x^4 + 2x^3 - 6x^2 + 8x - 3, & \text{if } x \in [1,2] \\ 1, & \text{if } x > 2 \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

and

$$f_{\min(X_1+X_2, X_3+X_4)}(x) = \begin{cases} -x^3 + 2x, & \text{if } x \in [0,1] \\ -x^3 + 6x^2 - 12x + 8, & \text{if } x \in [1,2] \\ 0, & \text{otherwise.} \end{cases}$$

$f_{\min(X_1+X_2, X_3+X_4)}$ and $F_{\min(X_1+X_2, X_3+X_4)}$ are depicted in Figures 4.3 and 4.4, respectively.

As the expected value of the cost of a proper hyperedge we get

$$E\left(\min\left(X_1+X_2, X_3+X_4\right)\right) = \int_{-\infty}^{\infty} x \cdot f_{\min(X_1+X_2, X_3+X_4)}(x)\,\mathrm{d}x = \frac{23}{30},$$

which is less than the mean value 1 of the uniform distribution in $[0,2]$. The smaller probability of values close to 0, however, seems to have a larger influence on the mean optimal value than the smaller mean cost as the computational results suggest.

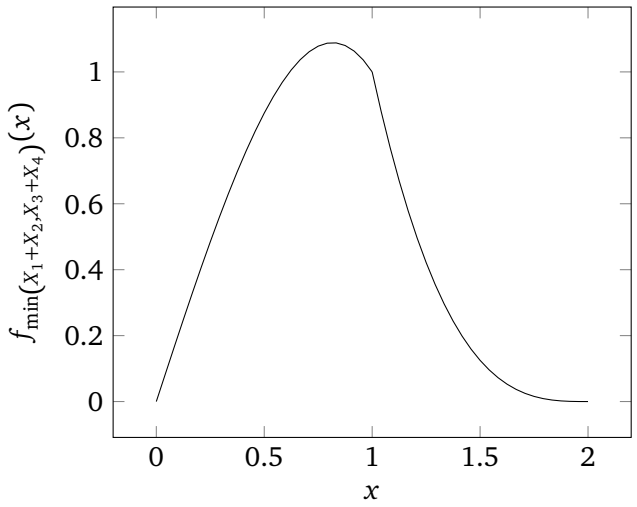


Figure 4.3: $f_{\min(X_1+X_2,X_3+X_4)}$ for the i. i. d. uniform random variables on $[0,1]$ X_1, X_2, X_3, X_4.

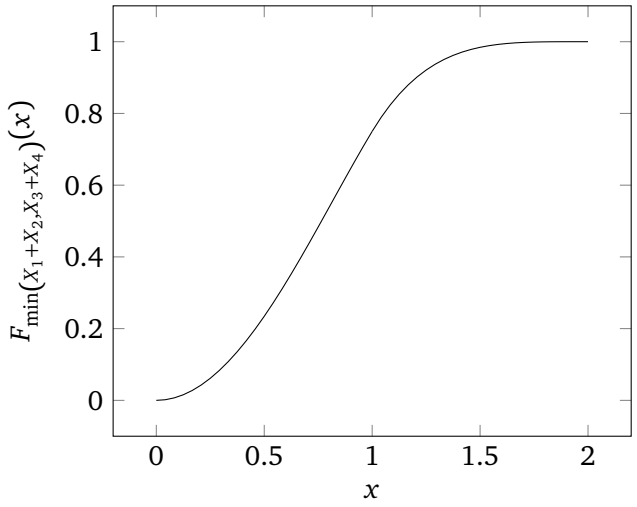


Figure 4.4: $F_{\min(X_1+X_2,X_3+X_4)}$ for the i. i. d. uniform random variables on $[0,1]$ X_1, X_2, X_3, X_4.

Chapter 5

Facet Classification without Normal Form—the Software "HUHFA"

The polyhedral approach to combinatorial optimization problems studies the structure of their associated polytopes. One way is to compute complete linear descriptions of small polytopes in order to generalize the equations and inequalities. "Small polytopes" might actually not look so small at first sight: There is often a huge number of facet-defining inequalities already for very small problem sizes.

However, there are often many symmetries implied by the combinatorial structure of the problem which can be used to classify the facets. These symmetries act on the feasible solutions and naturally form a group. In their representation as maps on the variable values they can be extended to symmetries acting on the polytope, and one can prove that they map vertices of the polytope to vertices of the polytope, and facets to facets. We say that those facet-defining inequalities which are similar in the sense that they can be transformed onto each other by some symmetry belong to one class.

Understanding all the facet-defining inequalities of a combinatorial optimization problem polytope then reduces to understanding one facet from each class.

To obtain this classification, one applies the symmetries to the facet-defining inequalities and then checks whether any two facets can be transformed into each other and hence belong to the same class. Often, this check is not so easy as two linear expressions describing the same facet might differ by the sum of multiples of several equalities from the problem description.

The check can be accomplished by defining a so-called normal form for the representation of inequalities—inequalities which have the same normal form describe the same facet. To this end, problem-specific normal forms were developed for some extensively studied combinatorial optimization problems. For the traveling salesman problem, every facet-defining inequality can be efficiently transformed to the so-called tight triangular normal form [Naddef and Rinaldi, 1993]. For an example of a normal form for the linear ordering problem, see [Reinelt, 1985]. In general, the representation of facet-defining inequalities in the orthogonal complement of the linear subspace spanned by the equations can be of course used as a normal form for the facets of a polytope. However, this needs techniques from linear algebra and can therefore raise numerical issues. Unfortunately, normal forms that can be described combinatorially are often not known. Hence, having a method that can be applied to every combinatorial optimization problem and relies solely on the combinatorial structure of the polytope is desirable.

Indeed, in this chapter we propose a novel technique, the algorithm "HUHFA", for classifying facets without using normal forms. The main idea is to identify every facet-defining inequality with the vertices of the polytope which satisfy it with equality. With this method, complete descriptions of polytopes computed by a software like PORTA [Christof and Loebel, 2008] (or a similar package) can be analyzed to divide the facets into equivalence classes according to groups generated by given symmetry mappings. It works regardless of whether the polytope is full-dimensional or not.

Facet classification methods without normal forms are also used in the Software SymPol [Rehn and Schürmann, 2010] for polyhedral representation transformations. To the best of our knowledge, their method relies on geometric scalar product invariants and algebraic invariants using polynomial rings as described in [Rehn, 2010, Section 3.2.2]. The invariant proposed here is much easier to compute and relies only on the vertex-facet incidence structure of the polytope.

This chapter is organized as follows. In Section 5.1, we present our approach for the classification of facet-defining inequalities. Section 5.2 gives some examples for symmetries, and in Section 5.3 we make a few comments about extensions of our theory for the classification of equations, which can be present in linear descriptions. Finally, we describe the implemented algorithm in Section 5.4 and give some computational results. The chapter closes with a discussion of the results obtained with HUHFA for the HAP in Section 5.5.

5.1 Equivalence of Facets

The following definition of symmetries will allow us to view facet classes as equivalence classes.

Definition 5.1.1. Let $s : x \mapsto Mx + r$ be a bijection on $\mathbb{R}^n$ with some (non-singular) matrix M and a vector r. The faces F_1 and F_2 of a polytope P are *equivalent with respect to* s if $s(\text{vert}(F_1)) = \text{vert}(F_2)$. If S is a set of bijections, then F_1 and F_2 are *equivalent with respect to* S if they are equivalent with respect to s for some $s \in S$. A bijection s is said to be a *symmetry* for P if $s(\text{vert}(P)) = \text{vert}(P)$.

The following two lemmas establish useful properties of a symmetry.

Lemma 5.1.2. *Let $s : x \mapsto Mx + r$ be a bijection on $\mathbb{R}^n$ with some (non-singular) matrix M and a vector r. Then $s^{-1} : x \mapsto M^{-1}x - M^{-1}r$.*

Proof. Since s is bijective, an inverse exists. We compute that $t(s(x)) = x$ for $t(x) = M^{-1}x - M^{-1}r$:

$$\begin{aligned} t(s(x)) &= M^{-1}(Mx + r) - M^{-1}r \\ &= x + M^{-1}r - M^{-1}r \\ &= x. \end{aligned}$$

□

Lemma 5.1.3. *Let P be a polytope and $s : x \mapsto Mx + r$ be a symmetry for P with some (non-singular) matrix M and a vector r. Then $s(P) = P$.*

Proof. Consider the $\mathcal{V}$-representation of P with $V := \text{vert}(P)$:

$$x = \sum_{v \in V} \lambda_v \cdot v, \ \sum_{v \in V} \lambda_v = 1, 0 \leq \lambda_v \leq 1 \ \forall v \in V$$

for every $x \in P$. Then,

$$\begin{aligned} s(x) &= s\left(\sum_{v \in V} \lambda_v \cdot v\right) \\ &= M \cdot \left(\sum_{v \in V} \lambda_v \cdot v\right) + r \\ &= M \cdot \left(\sum_{v \in V} \lambda_v \cdot v\right) + r \cdot \left(\sum_{v \in V} \lambda_v\right) \end{aligned}$$

$$= \sum_{v \in V} \lambda_v \cdot (Mv + r)$$
$$= \sum_{v \in V} \lambda_{s^{-1}(v)} v \in \text{conv}(V) = P.$$

Thus, $s(x) \in P$ for every $x \in P$.

Since $s^{-1}(x) = M^{-1}x - M^{-1}r$ is a symmetry, too, the argument from above proves also that $s^{-1}(x) \in P$ for every $x \in P$. $s(x) \in P$ and $s^{-1}(x) \in P$ for every $x \in P$ implies that $s(P) = P$. □

In our combinatorial understanding, a symmetry acts on the feasible solutions of a combinatorial optimization problem. This might seem not to match Definition 5.1.1 where a symmetry acts only on those feasible solutions which are vertices of the corresponding polytope. The vertices might be only a subset of the feasible solutions. However, this is not a restriction as feasible solutions that are vertices have to be mapped to feasible solutions that are vertices, too, by a symmetry in the combinatorial understanding. This is proven in the following lemma.

Lemma 5.1.4. *Let $s : x \mapsto Mx + r$ be a bijection on $\mathbb{R}^n$ with some (non-singular) matrix M and a vector r, and let P be a polytope. If $s(S) = S$ for some set $\text{vert}(P) \subseteq S \subseteq P$, then $s(\text{vert}(P)) = \text{vert}(P)$.*

Proof. We have to prove that $s(v) \in \text{vert}(P)$ for every $v \in \text{vert}(P)$.

Since v is a face, a valid inequality $a^T x \leq b$ exists for P such that $\{v\} = \{x : a^T x = b\} \cap P$. The inequality $a^T s^{-1}(x) \leq b \Leftrightarrow \left(a^T M^{-1}\right) x \leq aM^{-1}r + b$ is fulfilled by $s(v)$ with equality, and it is valid for $s(S)$, because $S \subseteq P$ and $a^T x \leq b$ is valid for P. On the other hand, for every $s(v')$ that fulfills $\left(a^T M^{-1}\right) x \leq aM^{-1}r + b$ with equality, $a^T v' = b$, so that v is the only point with this property. Thus, $\{s(v)\} = \left\{x : \left(a^T M^{-1}\right) x = aM^{-1}r + b\right\} \cap P$ is a vertex of P. □

We now show that a group of such symmetries induces indeed an equivalence relation on the faces of a polytope.

Lemma 5.1.5. *Let P be a polytope and S be a set of symmetries on P forming a group with respect to the composition operator ($\circ$). Then equivalence with respect to S defines an equivalence relation on the faces of P.*

Proof. Reflexivity holds because S contains the identity. Transitivity follows from the closure of S and the fact that if $\text{vert}(F_2) = s_1(\text{vert}(F_1))$ and $\text{vert}(F_3) = s_2(\text{vert}(F_2))$, then $\text{vert}(F_3) = (s_2 \circ s_1)(\text{vert}(F_1))$. Finally, symmetry follows from the existence of inverse elements in S. If $\text{vert}(F_2) = s(\text{vert}(F_1))$, then $\text{vert}(F_1) = s^{-1}(\text{vert}(F_2))$. □

Theorem 5.1.6. *Let F be a facet of the polytope P and $s : x \mapsto Mx+r$ a symmetry for P. If $a^T x \leq b$ defines the facet F, then $\left(a^T M^{-1}\right) x \leq b + a^T M^{-1} r$ defines some facet F' of P as well. Further, $s(F) = F'$ holds.*

Proof. Since s is bijective and by Lemma 5.1.3 $s^{-1}(x) \in P$ for every $x \in P$, the inequality $a^T s^{-1}(x) \leq b$ holds for every $x \in P$ as well. Written explicitly, we have $a^T(M^{-1}x - M^{-1}r) = a^T s^{-1}(x) \leq b$ and therefore the inequality $a^T M^{-1} x \leq b + a^T M^{-1} r$ is valid for P, too.

To complete the proof we will show that $\dim(F') = \dim(F)$ where

$$F' := \left\{x : \ a^T M^{-1} x = b + a^T M^{-1} r\right\} \cap P.$$

Because s is a bijection we get

$$\begin{aligned} a^T x = b &\Leftrightarrow a^T s^{-1}(s(x)) = b \\ &\Leftrightarrow a^T(M^{-1}s(x) - M^{-1}r) = b \\ &\Leftrightarrow (a^T M^{-1}) s(x) = b + a^T M^{-1} r \\ &\Leftrightarrow s(x) \in \left\{x : \ a^T M^{-1} x = b + a^T M^{-1} r\right\}. \end{aligned}$$

By Lemma 5.1.3 for s and s^{-1}, $x \in P \Leftrightarrow s(x) \in P$. Using the equivalence shown above, we get

$$\begin{aligned} x \in F &\Leftrightarrow x \in P \wedge a^T x = b \\ &\Leftrightarrow s(x) \in P \wedge a^T M^{-1} s(x) = b + a^T M^{-1} r \\ &\Leftrightarrow s(x) \in F'. \end{aligned}$$

Hence, $s(F) = F'$. Since s is an affine bijective map, it is dimension-preserving and the proof is completed. □

5.2 Examples and Groups of Bijections

In this section, we investigate some interesting groups of symmetries. We begin with examples of symmetries in two well-known combinatorial optimization problems. Afterwards, we prove possibilities to generate symmetry groups, and illustrate them using the hypercube as an example.

Contrary to the other problems described in this thesis, our examples here will be associated with *directed* graphs.

Example 5.2.1. The *complete directed graph* $D_n = (V_n, A_n)$ on n vertices has the vertex set $V_n = \{1, \ldots, n\}$ and a set of ordered pairs of vertices, called *arcs*,

$$A_n = \{(i, j) : \ i, j \in V_n\}.$$

A sequence $(v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k)$ with $k \geq 2$, $v_0 = v_k$, $v_i \in V$ for $i \in \{0, \ldots, k\}$ is called a *cycle* if for $i, j \in \{1, \ldots, k\}$ the edges $e_i \in E$ fulfill the condition $v_{i-1}, v_i \in e_i$ and the vertices fulfill the condition $v_i \neq v_j$ for $i \neq j$. It is called a *Hamiltonian cycle* or *tour* if $k = n$. For given arc length, the well-known *asymmetric traveling salesman problem (ATSP)* consists of finding a Hamiltonian cycle in D_n of minimum total length.

If a tour and a permutation σ of the vertices V_n are given, then the replacement of every arc (i, j) by $(\sigma(i), \sigma(j))$ yields again a tour. Furthermore, changing the direction of all arcs, in other words, replacing each arc (i, j) by (j, i), also gives a feasible tour. Thus, these two operations can be viewed as symmetries on the set of tours.

Consider the standard linear characterization of tours with binary variables x_{ij} where $x_{ij} = 1$ if arc (i, j) is in the tour, and $x_{ij} = 0$ otherwise.

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \qquad \forall j \in V_n \qquad \text{(TSP) (i)}$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \qquad \forall i \in V_n \qquad \text{(TSP) (ii)}$$

$$\sum_{i\in S} \sum_{j\in S, i\neq j} x_{ij} \leq |S| - 1 \qquad \forall S \subset V_n,\ 2 \leq |S| \leq n-1 \qquad \text{(TSP) (iii)}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A_n \qquad \text{(TSP) (iv)}$$

The *asymmetric traveling salesman polytope* $P^n(\text{ATSP})$ is defined as the convex hull of all feasible $0/1$ vectors of this characterization.

If σ is a permutation of V_n, then the map s^σ with $(s^\sigma(x))_{ij} = x_{\sigma^{-1}(i)\sigma^{-1}(j)}$ is an affine map. By applying s^σ to a feasible solution x we obtain the feasible solution x' where

$$x'_{ij} := \begin{cases} 1 & \text{if } x_{\sigma^{-1}(i)\sigma^{-1}(j)} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

The other symmetry, which reverses arcs, can be represented as the affine map r with $(r(x))_{ij} = x_{ji}$ converting a tour x to the tour x'' with

$$x''_{ij} := \begin{cases} 1 & \text{if } x_{ji} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Both maps s^σ and r are bijective. Further, they map the vertices of $P^n(\text{ATSP})$ onto themselves, so they are symmetries for $P^n(\text{ATSP})$.

For instance, consider the inequality $x_{12} + x_{13} + x_{14} \leq 1$, which is facet-defining for P^4(ATSP), and let σ be the permutation interchanging vertices 1 and 2. If we apply s^σ and r, we obtain the facet-defining inequality $x_{12} + x_{32} + x_{42} \leq 1$ that belongs to the same equivalence class.

Example 5.2.2. Let $D_n = (V_n, A_n)$ be the complete directed graph on n vertices. A *tournament* in D_n is a subset of A_n which contains for every pair (i, j) of vertices exactly one of the arcs (i, j) or (j, i). A tournament is acyclic if no subset of its arcs together with vertices from V_n is a cycle. For given arc weights, the well-known *linear ordering problem (LOP)* consists of finding an acyclic tournament of maximum total weight.

With binary variables y_{ij} indicating whether arc (i, j) is in the tournament or not, a linear characterization of acyclic tournaments is given by the system

$$\begin{aligned}
y_{ij} + y_{ji} &= 1 && \forall 1 \leq i < j \leq n && \text{(LOP) (i)}\\
y_{ij} + y_{jk} + y_{ki} &\leq 2 && \forall 1 \leq i, j, k \leq n,\ i < j,\ i < k,\ k \neq j && \text{(LOP) (ii)}\\
y_{ij} &\leq 1 && \forall 1 \leq i, j \leq n,\ j \neq i && \text{(LOP) (iii)}\\
y_{ij} &\in \mathbb{Z} && \forall (i, j) \in A_n && \text{(LOP) (iv)}
\end{aligned}$$

The *linear ordering polytope* P^n(LOP) is the convex hull of all feasible 0/1 vectors of this characterization.

As for the ATSP, also here vertex permutations and arc reversals can be used to convert acyclic tournaments into other acyclic tournaments. In addition to these symmetries, there is another type given in [Bolotashvili et al., 1999]. For a vertex $r \in V_n$ define the bijective map $\phi_r : \mathbb{R}^{n(n-1)} \to \mathbb{R}^{n(n-1)}$ by

$$(\phi_r(y))_{ij} = \begin{cases} y_{ji} & \text{if } i = r \text{ or } j = r \\ y_{ij} + y_{jr} - y_{ir} & \text{otherwise.} \end{cases}$$

This symmetry is a so-called "rotation mapping" and maps the linear ordering $1, 2, \ldots, r-1, r, r+1, \ldots n$ to the linear ordering $r+1, \ldots n, r, 1, 2, \ldots, r-1$. However, it is different from a vertex permutation symmetry: The interesting fact is that with this mapping the 3-dicycle inequalities (LOP) (ii) and the trivial inequalities (LOP) (iii) belong to the same equivalence class, which can be checked by an easy calculation.

In combinatorial optimization problems, variables often model whether some object is selected for the solution. Then, a permutation symmetry on these objects implies such a symmetry on the variables as stated in the next definition.

Definition 5.2.3. Let σ be a permutation of $\{1,\dots,n\}$. σ induces a linear bijective map $\overline{\sigma} : x = (x_1,\dots,x_n) \mapsto x_\sigma := (x_{\sigma(1)},\dots,x_{\sigma(n)})$.

Analogously, every subgroup S of the permutation group S_n defines a group $\overline{S} = \{\overline{\sigma} : \sigma \in S\}$ of bijections. In the following will not distinguish between σ and S_n and their implied bijections $\overline{\sigma}$ and $\overline{S_n}$ in notation.

We denote by $(i_1, i_2, \dots, i_k) \in S_n$ the permutation that maps i_j to $i_{(j \mod k)+1}$ for all $j = 1, 2, \dots, k$. For a group S and $s_1, s_2, \dots s_l \in S$, we write $< s_1, s_2, \dots, s_l >$ for the subgroup of S that is generated by $s_1, s_2, \dots, s_l$.

The following definition and the next lemma allow to create further groups of symmetries on a polytope.

Definition 5.2.4. Let $\tau : \mathbb{R} \to \mathbb{R}$ be an *involution*, i. e., $\tau \circ \tau = \mathrm{id}$. For a set $I \subseteq \{1,\dots,n\}$, we define the map $s_I^\tau : \mathbb{R}^n \to \mathbb{R}^n$ by setting

$$s_I^\tau(x)_i = \begin{cases} \tau(x_i) & \text{if } i \in I, \\ x_i & \text{otherwise.} \end{cases}$$

It is not hard to see that $s_I^\tau \circ s_I^\tau = \mathrm{id}$. In addition, define $S_J^\tau = \{s_I^\tau : I \subseteq J\}$ and $S^\tau = S_{\{1,\dots,n\}}^\tau$ when n is clear from the context.

Lemma 5.2.5. *For every $J \subseteq \{1,\dots,n\}$ and every involution $\tau : \mathbb{R} \to \mathbb{R}$, S_J^τ is a group of bijections with respect to composition.*

Proof. Because of $\emptyset \subseteq J$ it follows that $\mathrm{id} = s_\emptyset^\tau \in S_J^\tau$.

Existence of inverse functions is a direct consequence of s_I^τ being an involution.

Closure follows from the fact that for every $I_1, I_2 \subseteq J$ one has $I_1 \triangle I_2 \subseteq J$ (where $\triangle$ denotes symmetric difference) as well as $s_{I_1}^\tau \circ s_{I_2}^\tau = s_{I_1 \triangle I_2}^\tau$, which can be easily verified. □

Thus, for example, S_J^- with $- : x \mapsto -x$ and $J \subseteq \{1,\dots,n\}$ is a group of bijections with respect to composition and we could apply Lemma 5.1.5 to get an equivalence relation on a polytope for which the elements in S_J^- are symmetries.

Furthermore, we can show that compositions of certain types of bijections also form a group. They are defined as follows.

Definition 5.2.6. Let S and T be sets of bijections. The set $S \circ T$ is the set of all possible compositions of elements in S and T, i. e., $S \circ T = \{s \circ t : s \in S, t \in T\}$.

The following two lemmas state that we can compose S_n and S^- to generate new groups of bijections.

Lemma 5.2.7. *Let S, T be groups. $S \circ T$ is a group if and only if $S \circ T = T \circ S$.*

Proof. **Part 1: $S \circ T = T \circ S \Rightarrow S \circ T$ is a group.**

(i) Because $\mathrm{id} \in S, T$, we can write $\mathrm{id} = (\mathrm{id} \circ \mathrm{id}) \in S \circ T$.

(ii) To show the existence of inverse elements let $(s \circ t) \in S \circ T$ with $s \in S$ and $t \in T$. The inverse of $(s \circ t)$ is clearly given by $(t^{-1} \circ s^{-1})$ which is in $T \circ S = S \circ T$ again.

(iii) $S \circ T = T \circ S$ means that for any $(t \circ s) \in T \circ S$, we have $(t \circ s) = (s' \circ t')$ for some $s' \in S, t' \in T$ and vice versa (*). Using this rule, we can show that

$$\begin{aligned}(s_1 \circ t_1) \circ (s_2 \circ t_2) &= (s_1 \circ t_1) \circ (t_3 \circ s_3) \\ &= s_1 \circ t_1 \circ t_3 \circ s_3 \\ &= s_1 \circ ((t_1 \circ t_3) \circ s_3) \\ &= s_1 \circ (s_4 \circ t_4) \\ &= (s_1 \circ s_4) \circ t_4 \in S \circ T.\end{aligned}$$

The substitutions are all due to (*). This shows composition.

Part 2: $S \circ T$ is a group $\Rightarrow S \circ T = T \circ S$. It suffices to show that for any $s \in S, t \in T$, we have $(t, s) \in S \circ T$ and $(s, t) \in T \circ S$. First, consider $\left(s^{-1} \circ t^{-1}\right) \in S \circ T$. Because $S \circ T$ is a group, it follows for the inverse that $(t \circ s) = \left(s^{-1} \circ t^{-1}\right)^{-1} \in S \circ T$, which shows $T \circ S \subseteq S \circ T$.

We proceed to show that $T \circ S$ is also a group, so swapping S and T in the argument before shows $T \circ S \supseteq S \circ T$ and we are done.

Reflexivity is clear since $(\mathrm{id} \circ \mathrm{id}) \in T \circ S$. To see the closure, choose some $s, s' \in S$ and $t, t' \in T$. By the closure of $S \circ T$,

$$\left(s'^{-1} \circ t'^{-1}\right) \circ \left(s^{-1} \circ t^{-1}\right) = \left(s''^{-1} \circ t''^{-1}\right)$$

holds for some $s'' \in S, t'' \in T$. Its inverse reads as

$$(t \circ s) \circ \left(t' \circ s'\right) = \left(t'' \circ s''\right).$$

This proves the closure of $T \circ S$.

For $(t \circ s) \in T \circ S$, it follows by closure that

$$(t \circ s)^{-1} = \left(s^{-1} \circ t^{-1}\right) = \left(\mathrm{id} \circ s^{-1}\right) \circ \left(t^{-1} \circ \mathrm{id}\right) \in T \circ S,$$

which shows the existence of inverse elements and completes the proof. □

Lemma 5.2.8. *For the groups $S \subseteq S_n$ and S^τ, we have $S_n^\tau := S \circ S^\tau = S^\tau \circ S$ and S_n^τ is a group.*

Proof. It suffices to show explicitly that $\sigma \circ s_I^\tau = s_{\sigma(I)}^\tau \circ \sigma$ for $\sigma \in S$ and $s_I^\tau \in S^\tau$. Then we can apply Lemma 5.2.7.

For $x \in \mathbb{R}^n$ we have

$$(\sigma \circ s_I^\tau)(x)_i = \begin{cases} \tau(x_{\sigma(i)}) & \text{if } i \in I \\ x_{\sigma(i)} & \text{otherwise} \end{cases}$$

and

$$(s_{\sigma(I)}^\tau \circ \sigma)(x)_i = \begin{cases} \tau(x_{\sigma(i)}) & \text{if } \sigma(i) \in \sigma(I) \\ x_{\sigma(i)} & \text{otherwise.} \end{cases}$$

Since σ is a bijection, $i \in I \iff \sigma(i) \in \sigma(I)$, which shows that the two functions above are identical and thus $S \circ S^\tau = S^\tau \circ S$. □

Example 5.2.9. We define the involution

$$\neg : x \mapsto 1 - x.$$

If we view 0 and 1 as Boolean values, $s^\neg$ simply negates them. By Lemma 5.2.5, $S^\neg = \{s_I^\neg : I \subseteq \{1, \dots, n\}\}$ forms a group and defines equivalence classes on the faces of any polytope P it is a symmetry for. Further, $S_n^\neg := S_n \circ S^\neg = S^\neg \circ S_n$ is a group by Lemma 5.2.7.

Consider the so-called *hypercube* $Q_n \subset \mathbb{R}^n$ given by its $\mathcal{V}$-representation

$$Q_n = \operatorname{conv}(\{0,1\}^n).$$

The vertices of Q_n are $\{0,1\}^n$. The facets of Q_n are defined by the inequalities $-x_i \leq 0$ and $x_i \leq 1$ for $i = 1, \dots, n$. We get the following different equivalence classes with respect to the different symmetry groups S_n, $S^\neg$ and $S_n^\neg$.

S_n : As we can swap any two variables x_i and x_j with each other but cannot do anything about the form of the inequality, we get exactly two equivalence classes:

- $\{\{x \in P : x_i = 0\} : i \in \{1, \dots, n\}\}$,
- $\{\{x \in P : x_i = 1\} : i \in \{1, \dots, n\}\}$.

See Figures 5.1–5.3 for a visualization.

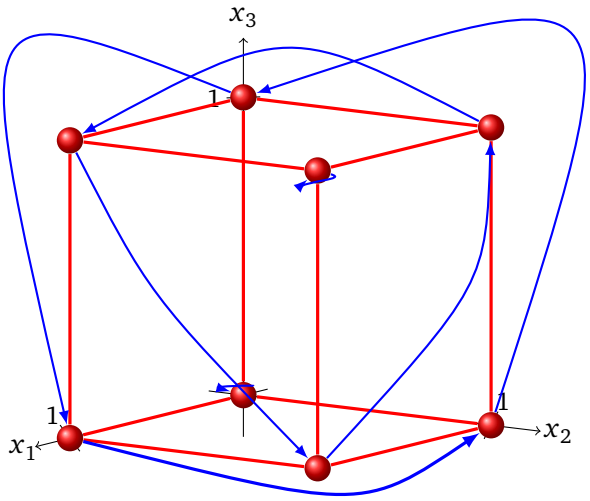


Figure 5.1: This figure shows where the symmetry $(x_1, x_2, x_3) \mapsto (x_2, x_3, x_1)$ maps the vertices of the cube.

$S^{\neg}$: Using $s^{\neg}_{\{i\}}$ one can transform the facet $\{x \in P : -x_i = 0\}$ into the facet $\{x \in P : x_i = 1\}$, but we cannot change which variables occur in the inequality. So we get a total of n equivalence classes, one for each $i = 1, \ldots, n$ of the form:

$$\{\{x \in P : -x_i = 0\}, \{x \in P : x_i = 1\}\}$$

See Figure 5.4 for a visualization.

$S^{\neg}_n$: As we can swap any two variables x_i, x_j as well as use $s^{\neg}_{\{i\}}$ to transform the facet-defining inequalities into one another, we only have one equivalence class which includes every facet of Q_n. Given the symmetric nature of Q_n, this is exactly what we would like to see if we are interested in geometric properties.

This example shows that the equivalence classes depend on the chosen symmetry group.

5.3 Equivalence of Equations

Many combinatorial polytopes require equations for their linear description. However, we unfortunately cannot proceed in the same way as for inequalities

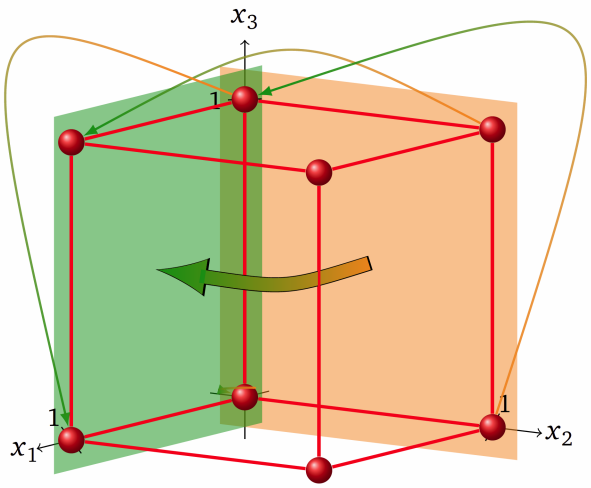


Figure 5.2: The symmetry $(x_1, x_2, x_3) \mapsto (x_2, x_3, x_1)$ maps the facet of the cube defined by $x_1 \geq 0$ onto the facet defined by $x_2 \geq 0$.

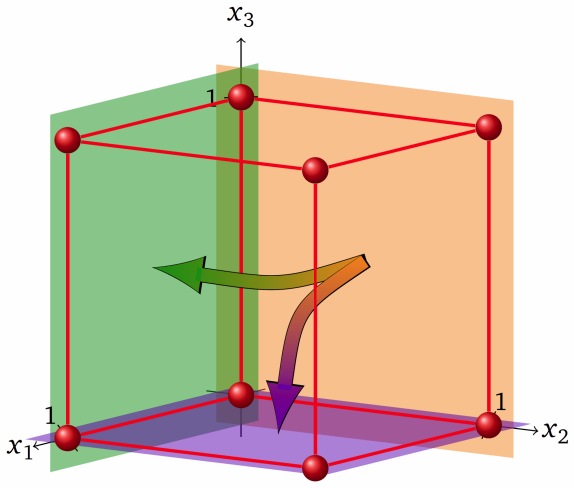


Figure 5.3: Shows where permutation symmetries could map the facet of the cube defined by $x_1 \geq 0$.

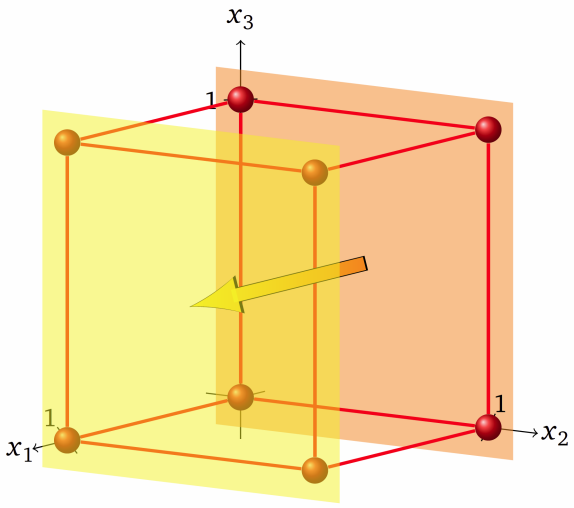


Figure 5.4: The symmetry $(x_1, x_2, x_3) \mapsto (1 - x_1, x_2, x_3)$ maps the facet of the cube defined by $x_1 \geq 0$ onto the facet defined by $x_1 \leq 1$. It cannot be mapped to this facet by a permutation symmetry.

to classify them. The reason is that the vertices are not an invariant of equations since every point in P has to satisfy them.

After an example of what classification of equations means for the asymmetric traveling salesman problem, we will present a theoretical analysis of different invariants related to equations and group of symmetries.

Example 5.3.1. In Example 5.2.1, we analyzed the inequalities in the $\mathscr{H}$-representation of the asymmetric traveling salesman polytope. However, there are also n Equations (TSP) (i) and n Equations (TSP) (ii) present in the description of the polytope.

With the same symmetries as in Example 5.2.1, vertex permutations s^σ and the arc reversal r, these $2n$ equations belong to one equivalence class: Using the vertex permutation σ that changes the vertices $i_1 \in V_n$ and $i_2 \in V_n$, any two Equations (TSP) (i) with $i = i_1$ and $i = i_2$ belong to the same class. In the representation given, we can see this just by mapping the variables using the permutation symmetry σ. The same holds for the Equations (TSP) (ii) by interchanging the vertices $j = j_1$ and $j = j_2$. Further, arc reversals imply that Equations (TSP) (i) and (TSP) (ii) for the same vertex $i = j$ belong to the same class. Thus, the n Equations (TSP) (i) and n Equations (TSP) (ii) all belong to the same equivalence class.

However, there are many other equivalent representations of Equations

(TSP) (i) and (TSP) (ii). For example, the given equations could be iteratively changed by adding a multiple of some equation to another one. Although this would of course not change the polytope, it would be not as easy as above to see that the equations belong to the same equivalence class.

We will now study different approaches to classify equations that might be given in different representations. Our results show that the simpler approaches do not lead to valid equivalence relations.

As a starting point, the next definition introduces the concept of a linear subspace that represents all the equations that can be generated by applying all symmetries from a given group to one equation. We will need this notion in our exploration of possible invariants for the identification of equations regardless of their representation.

Definition 5.3.2. Let S be a group of symmetries that are bijections on $\mathbb{R}^n$ and $c^T x = d$ a valid equation for the polytope $P \subseteq \mathbb{R}^n$. We can identify the equation with the vector $(c,d) \in \mathbb{R}^{n+1}$. Then the *symmetric subspace* $U_S(c,d) \subset \mathbb{R}^{n+1}$ of (c,d) over S is defined as

$$U_S(c,d) := \text{span}(\{(c_\sigma, d_\sigma) : \ \sigma \in S, \sigma(x) = M_\sigma x + r_\sigma\}),$$

where $c_\sigma := M^T c$ and $d_\sigma = d - c^T r$.

The following Lemma establishes an equivalence relation on equations.

Lemma 5.3.3. *The relation*

$$(c,d) \simeq_1 (c',d') :\Leftrightarrow U_S(c,d) = U_S(c',d')$$

is an equivalence relation on the valid equations of a polytope P.

Proof. Being an equivalence relation is a direct consequence of "=" being one:

- $(c,d) \simeq_1 (c,d)$ since $U_S(c,d) = U_S(c,d)$,
- $(c,d) \simeq_1 (c',d') \Rightarrow (c',d') \simeq_1 (c,d)$ since

$$U_S(c,d) = U_S(c',d') \Rightarrow U_S(c',d') = U_S(c,d),$$

- $(c,d) \simeq_1 (c',d'),\ (c',d') \simeq_1 (c'',d'') \Rightarrow (c,d) \simeq_1 (c',d')$ since

$$U_S(c,d) = U_S(c',d'),\ U_S(c',d') = U_S(c'',d'') \Rightarrow U_S(c,d) = U_S(c'',d'').$$

□

Lemma 5.3.3 shows an equivalence relation on equations. But $\simeq_1$ does not necessarily produce a minimal number of equivalence classes as we will show now. Consider the following example.

$$\begin{aligned} S &:= < (2,3),(4,5) > \subseteq S_5, \\ (c,d) &:= (1,1,-1,0,0,0), \\ (c',d') &:= (1,0,0,0,0,0), \\ (c'',d'') &:= (1,0,0,1,-1,0). \end{aligned}$$

Using the equivalence relation $\simeq_1$, we get three classes because we have $(c,d) \not\simeq_1 (c',d') \not\simeq_1 (c'',d'') \not\simeq_1 (c,d)$.

However, we might want to view (c,d), (c',d') and (c'',d'') as belonging to one equivalence class w. r. t. S. For each pair of these vectors $(c^1,d^1),(c^2,d^2)$, one of them lies in the symmetric subspace of the other one, i. e., $(c^1,d^1) \in U_S(c^2,d^2)$ or $(c^2,d^2) \in U_S(c^1,d^1)$ holds.

Therefore, the result in the last lemma is not satisfying. But simpler approaches do not lead to an equivalence relation, as we will show in the remainder of this section.

But at least we can use this approach to obtain some subclasses where all members are equivalent. As mentioned above, these classes are not necessarily complete.

For practical computations, we consider this result as not strong enough to be worth implementing. However, this is not such a big problem since most polytopes do not contain that many equations and their number is bounded by the number of variables.

The following lemma leads to further ideas for equivalence relations on equations.

Lemma 5.3.4. *The following implication is true:*

$$(c,d) \in U_S(c',d') \Rightarrow U_S(c,d) \subseteq U_S(c',d').$$

Proof. $(c,d) \in U_S(c',d')$ means that there is a representation

$$(c,d) = \sum_{\sigma \in S} \lambda_\sigma (c'_\sigma, d'_\sigma)$$

with $\lambda_\sigma \in \mathbb{R}$. Now consider for $\tau \in S$ that $(c_\tau, d_\tau) \in U_S(c,d)$. Using τ on the

representation of (c,d) in $U_S(c',d')$ yields

$$\begin{aligned}(c_\tau, d_\tau) &= \sum_{\sigma\in S} \lambda_\sigma((c'_\sigma)_\tau, (d'_\sigma)_\tau)\\ &= \sum_{\sigma\in S} \lambda_{(\tau^{-1}\circ\tau\circ\sigma)}(c'_{\tau\circ\sigma}, d'_{\tau\circ\sigma})\\ &= \sum_{\sigma\in\tau(S)} \lambda_{(\tau^{-1}\circ\sigma)}(c'_\sigma, d'_\sigma)\\ &= \sum_{\sigma\in S} \lambda_{(\tau^{-1}\circ\sigma)}(c'_\sigma, d'_\sigma) \in U_S(c',d').\end{aligned}$$

□

Unfortunately, the implication in Lemma 5.3.4 does not result in an equivalence relation as we show now.

Lemma 5.3.5. *The relation*

$$(c,d) \simeq_2 (c',d') :\Leftrightarrow (c,d) \in U_S(c',d')$$

is not an equivalence relation on the valid equations of a polytope P.

Proof. Although the relation is reflexive and transitive, it may not be symmetric; a counterexample is easy to construct:

$$S :=< (2,3) >\subseteq S_3, \qquad (c,d) := (1,0,0,0), \qquad (c',d') := (1,1,-1,0).$$

□

To avoid the problem that the relation is not symmetric, we try a weaker approach.

Lemma 5.3.6. *The relation*

$$(c,d) \simeq_3 (c',d') :\Leftrightarrow (c,d) \in U_S(c',d') \text{ or } (c',d') \in U_S(c,d)$$

is not an equivalence relation on the valid equations of a polytope $P \subset \mathbb{R}^n$.

Proof. Although the relation is reflexive and symmetric, it may not be transitive. A counterexample where $(c,d) \simeq_3 (c',d'), (c',d') \simeq_3 (c'',d'')$ but $(c,d) \not\simeq_3 (c'',d'')$ is again the one we used in the discussion of the result in Lemma 5.3.3,

$$\begin{aligned}S &:=< (2,3),(4,5) >\subseteq S_5,\\ (c,d) &:= (1,1,-1,0,0,0),\\ (c',d') &:= (1,0,0,0,0,0),\\ (c'',d'') &:= (1,0,0,1,-1,0).\end{aligned}$$

□

So this approach also does not work. The only possibility seems to be to define an equivalence relation based on Definition 5.3.2 and to demand the equivalence of the two spans themselves as shown in Lemma 5.3.3.

5.4 A Facet Classification Algorithm

Based on the results of the previous sections we propose Algorithm 5.4.1, HUHFA, for facet classification, see page 70.

Theorem 5.4.1. *Algorithm 5.4.1 works correctly and terminates.*

Proof. We first prove that the algorithm is well-defined. It is obvious that it then also terminates.

The only critical part is Step 20 where it is not immediately clear that PermIndex is the incidence vector of a facet of P. However, because the s_j is a symmetry for P, this property follows directly from Theorem 5.1.6.

Thus, the proof of termination is completed and we proceed with the correctness. We will need the following definition. Let $G_{P,S} = (V, E)$ denote the graph of facet symmetries of P with regard to S, i. e., $V = \{1, ..., m\}$ where $i \in V$ refers to the facet F_i induced by f_i, and $\{i, j\} \in E$ if and only if there exists a map $s \in S$ such that either f_i is equivalent to f_j with regard to s or vice versa. Note that we allow cycles to arise.

To show correctness, we first prove that the algorithm computes the connected components of $G_{P,S}$ such that the corresponding facet-defining inequalities from one component all have the same Class in the algorithm.

Suppose that S is closed. If we have both edges $\{a, b\}$ and $\{b, c\}$ in E it follows from closure that we have the edge $\{a, c\}$ as well. Using this argument iteratively on every path connecting vertices i and j shows that already $\{i, j\} \in E$ which implies that the connected components form complete subgraphs. In this case, we get the whole connected component of some $i \in V$ by examining all its incident edges, which is exactly what the algorithm does in Steps 15–21.

So let S be not closed. This means that we have to check the neighborhood of every vertex in order to compute the connected components, which again is exactly what the algorithm does in Steps 15–21.

Therefore, the main argument for correctness in the case in which S is not closed lies in the following claim: $G_{P,S}$ and $G_{P,<S>}$ have the same connected components.

We may view the maps in S as permutations of the set vert(P) where their explicit form as affine maps is only used to encode these vertex permutations. (In fact, this is what the algorithm does in Steps 12–14. However, saving the

Algorithm 5.4.1: HUHFA.

Data: Min. $\mathscr{H}$, $\mathscr{V}$-representations of polytope $P \subset \mathbb{R}^n$ and symmetries given by
- list of vertices $v_1, v_2, \ldots, v_k$,
- list of facet-defining inequalities $f_1, f_2, \ldots, f_m$,
- set $S = \{s_1, s_2, \ldots, s_l\}$ of symmetries on P,
- Boolean closed indicating whether $(S, \circ)$ is a group.

Result: Equivalence classes Class of facets of P w. r. t. $< S >$.

```
1  Incidence ← two-dimensional array of size m × k
2  P ← two-dimensional array of size l × k
3  Class ← array of size m
4  Index ← empty list of key-value pairs (array with k entries (incidence vector of
   facet), integer) lexicographically sorted by the keys
5  VertIndex ← list of key-value pairs (vertex v_i, i) lexicographically sorted by the
   keys
6  PermIndex ← array of size k
7  for i ← 1 to m do                    // for all facet-defining inequalities
8      for o ← 1 to k do                                  // for all vertices
9          Incidence[i,o] ← 1 if v_o satisfies the inequality f_i with equality, 0
           otherwise                                // create incidence vector
10     Index( Incidence[i])←i          // index the vector to identify it
       quickly
11     Class[i]←i                              // set class of facet f_i to i
12 for j ← 1 to l do                                    // for all symmetries
13     for o ← 1 to k do                                  // for all vertices
14         P[j,o]← VertIndex(s_j(v_o))                 // create mapping table
15 for i ← 1 to m do                  // for each facet-defining inequality
   f_1, f_2, ..., f_m
16     if not closed or Class[i]=i then
17         for j ← 1 to l do                  // for each symmetry s_1, s_2, ..., s_l
18             for o ← 1 to k do                  // for each vertex v_1, v_2, ..., v_k
19                 PermIndex[o]=Incidence[i,P[j,o]]     // create vector by
                   using j on the incidence vector of j
20             ImageIndex ← Index(PermIndex)      // identify class
21             unite classes i and ImageIndex in Class                // with a
               disjoint-set data structure maintained and updated
               in parallel to Class
22 return Class
```

maps as vertex permutations is not necessary but is supposed to reduce the running time.) Then we can view $< S >$ as a finite group. Finiteness is justified because we retain all the information needed to determine the edge sets.

Obviously, because $S \subseteq < S >$, every pair of vertices connected in $G_{P,S}$ is connected in $G_{P,<S>}$ as well. So suppose that $i \in V$ and $j \in V$ are connected in $G_{P,<S>}$. As stated previously, the connected components of $G_{P,<S>}$ form complete subgraphs such that $\{i, j\}$ is an edge in $G_{P,<S>}$. By our definition, this means that there exists a map $s_{ij} \in < S >$ such that F_i and F_j are equivalent with respect to s_{ij}.

By definition of $< S >$ this means that s_{ij} can be written as the composition of a sequence of maps from S. Because $< S >$ is finite, we can assume this sequence to be finite as well. Suppose that $s_{ij} = \sigma_t \circ \ldots \circ \sigma_2 \circ \sigma_1$ where $\sigma_r \in S$ for $r = 1, \ldots, t$. By Theorem 5.1.6, we have that $F^{(r)} := (\sigma_r \circ \ldots \circ \sigma_2 \circ \sigma_1)(F_i)$ is a facet of P as well. In this setting $F^{(t)} = F_j$, $F^{(0)} = F_i$ and we have that $F^{(r+1)} = \sigma_{r+1}(F^{(r)})$ for $r = 0, \ldots, t-1$. But now it follows from our definition of the edge set that this does in fact define a path connecting i and j in $G_{P,S}$. Therefore the claim and the theorem are proven. □

Theorem 5.4.2. *Let c denote the number of equivalence classes with respect to S. Then Algorithm 5.4.1 has a worst case running time of*

$$O(m(nk + \log(m)) + kl(n^2 + \log(k)) + ml(k + \log(m)))$$

if S is not closed, and

$$O(m(nk + \log(m)) + kl(n^2 + \log(k)) + cl(k + \log(m)))$$

if S is closed.

Proof. Running time of Steps 1–6 can be neglected.

Loop 7 has m iterations, Loop 8 has k iterations. Step 9 consists of at most n multiplications and $n-1$ addition operations. Step 10 adds an element to a sorted list with $\leq m$ entries, this is in $O(\log(m))$. Step 11 is in $O(1)$. Thus, Steps 7–11 are in $O(m(kn + \log(m)))$.

Loops 12 and 13 have l and k iterations, respectively. Step 14 involves the evaluation of an affine map, which can be done in $O(n^2)$, and finding an element in a sorted list with k entries, which can be done in $O(\log(k))$. All other operations in this step are in $O(1)$. Thus, Steps 12–14 have a running time in $O(kl(n^2 + \log(k))$.

There are m iterations of Loop 15, l iterations of Loop 17 and k iterations of Loop 18. Step 19 is in $O(1)$ and Step 20 in $O(\log(m))$. Because of the use of

a disjoint-set data structure, Step 21 has an amortized constant of $\alpha(m)$, which denotes the inverse of the Ackermann function (this value does not rise above 4 for all practical purposes, [Cormen et al., 1989]).

If S is closed, we enter Step 21 exactly once for every class, and we get the estimate of $O(cl(k+\log(m)+\alpha(m))) = O(cl(k+\log(m)))$ for Steps 15–21 where c denotes the number of equivalence classes with respect to S. Otherwise we do Step 21 once for every facet and get $O(ml(k+\log(m)+\alpha(m))) = O(ml(k+\log(m)))$ for these steps.

All together, these terms sum up to

$$O(m(nk+\log(m)) + kl(n^2+\log(k)) + al(k+\log(m)))$$

where $a = c$ if S is closed and $a = m$ otherwise.

□

With reasonable assumptions the term can be simplified:

Corollary 5.4.3. *Assuming* $\log(m) \leq k \leq m$, $n \leq k$, $n \in O(m)$, *sparsity of the affine maps in S, that is, they can be applied to the vertices in linear time, and* $|S| \in O(k)$, *Algorithm 5.4.1 has a running time in* $O(mk^2)$.

These assumptions are reasonable in the sense that they are often encountered when dealing with combinatorial problems and their LP-formulations. In this scenario, symmetries are often given by permuting variables or flipping their respective 0/1-values, which naturally results in sparse matrices. At the same time, it is a well known fact that every subgroup of the symmetric group S_k can be generated by at most k maps, and because $< S >$ is a subgroup of S_k (it permutes k vertices), this applies to $< S >$ as well. Thus generators can be chosen accordingly, and there are known algorithms for this task which run in polynomial time [Jerrum, 1986]. The difficulty would be to find a low number of generators which possess a sparse affine map presentation at the same time but as said before, this case arises naturally in practice. The estimates for m and k and their relation to n are typical for $\mathcal{NP}$-hard problems.

With an implementation of Algorithm 5.4.1 we were able to produce the results shown in Table 5.1. The algorithm was implemented in C++ and all computations were performed on an Intel(R) Core(TM) i7-2600 CPU 4-core processor with 3.40 GHz and 16 GB RAM.

With our computations, we contributed to the understanding of the polyhedral structure of certain combinatorial optimization problems with a large number of facets. Although we could calculate all the facet-defining inequalities from the known $\mathcal{V}$-descriptions before, classifying the inequalities manually

polytope	#var.	#vert.	#facets	$\|S\|$	$\|< S >\|$	closed	#cla.	CPU
P^6(LOP)	15	720	910	6	??	false	2	2.2 s.
P^5(TSP)	20	24	390	5	240	false	6	0.8 s.
P^4(TVP)	18	24	1280	4	48	false	48	0.2 s.
P^5(STVP)	30	120	30040	5	240	false	175	17.2 s.
P^3(HAP)	45	978	14049	4608	4608	true	30	90.3 s.

Table 5.1: Running times of HUHFA for different combinatorial optimization problem polytopes. For each polytope the number of variables, vertices, facets can be found in the table. Further, cardinality of the set of symmetries S, the cardinality of $< S >$ if known, whether S is closed, the number of equivalence classes in the result and the running time of HUHFA is shown.

was not possible. Further, we could show that Algorithm 5.4.1 results in a practically usable software.

As test instances we used several LOP and TSP polytopes as well as polytopes describing the target visitation problem [Hildenbrandt et al., 2013], a combination of the linear ordering problem and the traveling salesman problem, in a symmetric and asymmetric version (STVP/TVP), or the HAP in $G_{2,3}$. Both TVP, STVP and HAP are problems where already very small instances are described by a very large number of facets.

Table 5.1 shows the computation times of our algorithm for these problems. The chosen problem sizes are the maximum ones which PORTA is able to compute in a reasonable amount of time. As you can see, HUHFA can classify the facets for the different problems that we have included in our computations in a practically acceptable time.

The software HUHFA can be downloaded from the website `http://comopt.ifi.uni-heidelberg.de/people/hildenbrandt/HUHFA`.

5.5 Example: HUHFA for the HAP in $G_{2,3}$

In this section, we will discuss the results obtained with HUHFA for the HAP polytope P(HAP).

The largest input hypergraph for which we were able to compute its facets using PORTA (with a running time of about two weeks on an Intel(R) Xeon(R) CPU E3-1290 V2 4-core processor with 3.70 GHz and 16 GB RAM—the number

of processors, however, does not matter as PORTA is not parallelized in the current version) was the complete partitioned input hypergraph $G_{2,3}$ with three parts on the U- and V-side, all of size two. We enumerated all the 978 vertices of the corresponding polytope and PORTA found that it has 14049 facets. Our aim was to get a combinatorial understanding of the facets in order to be able to generalize them to HAP polytopes for larger hypergraphs. Over 14 thousand facets is, however, an enormous number to deal with.

We therefore used HUHFA to classify the facets into equivalence classes so that we would have to analyze only one facet from each class. As symmetries, we used all those that are generated by interchanging U and V, permuting the parts on one side, and interchanging the two vertices from one part. To state this more formally, denote for $G_{2,n} = (U, V, E)$ by $U_1 = \{u_{11}, u_{12}\}$, ..., $U_n = \{u_{n1}, u_{n2}\}$ the n parts on the U-side, and by $V_1 = \{v_{11}, v_{12}\}$, ..., $V_n = \{v_{n1}, u_{n2}\}$ the n parts on the V-side. A permutation σ of the vertices $U \cup V$ of $G_{2,n}$ defines a symmetry s_σ on $\mathbb{R}^E$ that can be written as follows.

$$(s_\sigma(x))_e = x_{\{\sigma(v):\ v \in e\}}$$

The symmetry that interchanges U and V can be stated using the vertex permutation $\sigma_1 : U \cup V \to U \cup V$,

$$\sigma_1 : u_{ij} \mapsto v_{ij}, v_{ij} \mapsto u_{ij}.$$

For some permutation τ of $\{1, \ldots, n\}$, we can define a part permutation in U using $\sigma_2 : U \cup V \to U \cup V$,

$$\sigma_2 : u_{ij} \mapsto u_{\tau(i)j}, v_{ij} \mapsto v_{ij}.$$

Finally, a symmetry that interchanges the two vertices in the part U_1 can be deduced from the vertex permutation $\sigma_3 : U \cup V \to U \cup V$,

$$\sigma_3 : u_{11} \mapsto u_{12}, u_{12} \mapsto u_{11}, u_{ij} \mapsto u_{ij} \text{ for all } i \neq 1, v_{ij} \mapsto v_{ij}.$$

HUHFA returned that the 14049 facets of $G_{2,3}$ belong to 30 facet classes with respect to the 4608 symmetries generated by $s_{\sigma_1}, s_{\sigma_2}$ for different permutations τ that generate S_n, and s_{σ_3}. One facet from each equivalence class is listed in Appendix B.

We found that all the facet-defining inequalities for these facets can be written with coefficients -1, 0 and 1, and a right hand side equal to 1. We do not know, however, whether this result holds in general for HAP polytopes. We found a different representation, namely one with only non-negative coefficients, more helpful to understand the combinatorial meaning of certain facets.

Two of the facet classes correspond to the non-negativity constraints (HAP) (ii). They belong to two equivalence classes since edges and the proper hyperedges of size 4 cannot be mapped to each other by the symmetries. One facet class represents the only type of cliques in $G_{2,3}$ (and also $G_{2,n}$ in general). 13 further classes can be interpreted as a generalization of odd set inequalities for the matching problem as we will show in Section 6.3. The 14 other facet classes remain uninterpreted.

It was not possible for us to study the polytope of the set packing and set covering relaxation of the HAP in this way since the polytopes P_{LP}(SSP) and P_{LP}(SCP) for $G_{2,3}$ have a much higher number of vertices than P_{LP}(SPP). The running time of PORTA would have been unreasonably long.

Chapter 6

Polyhedral Results and Dual Methods

This chapter is about polyhedral results for the HAP. We begin with Section 6.1 on the dimension of the HAP polytope $P(\text{HAP})$. Then, in Section 6.2, we will introduce an extended IP formulation of polynomial size and study its projection to the variables of the canonical HAP formulation. As a corollary, we will see that it implies all clique inequalities. As another polyhedral result, we will generalize the odd set inequalities for the matching problem to feasible inequalities for the set packing problem (and therefore also the HAP) in Section 6.3. Some of these inequalities are facets for $G_{2,3}$ which were calculated and classified using HUHFA, see Section 5.5. We will also relate them to another generalization of odd set inequalities published in [Pêcher and Wagler, 2006], and show how both generalizations can be combined.

6.1 Dimension

In this section, we will explore the dimension of the HAP polytope $P(\text{HAP})$. Since the HAP is $\mathcal{NP}$-hard (see Chapter 3), even the question whether $P(\text{HAP})$ is empty, i. e., its dimension is equal to -1, is $\mathcal{NP}$-complete for general bipartite hypergraphs. However, we can state the dimension at least for bipartite hypergraphs that include all possible edges in their hyperedge sets. To prove this we will extend the dimension result for the polytope of the assignment problem in a complete bipartite graph $G = (U, V, E)$. It can be shown that its dimension is $|E| - 2 \cdot |V| + 1$ [Lovász and Plummer, 1986].

Theorem 6.1.1. *Let $G = (U, V, E)$ be a bipartite hypergraph such that $E_1 := \{(u, v) : u \in U, v \in V\} \subseteq E$. Then the dimension of P(HAP) for G is $|E| - 2 \cdot |V| + 1$.*

Proof. We first show that the dimension cannot be greater than the stated value. As the IP formulation (HAP) has $|E|$ variables, we have to show that its equation system (HAP) (i) contains $2 \cdot |V| - 1$ linearly independent equations. We will do this by calculating the column rank of the corresponding coefficient matrix. A column corresponding to a proper hyperedge is the sum of columns corresponding to a set of edges whose union is the hyperedge (this always exists since E contains all possible edges), and the column rank for the submatrix with columns corresponding to edge variables is $2 \cdot |V| - 1$ [Lovász and Plummer, 1986].

It is left to show that the dimension is at least the stated value. This can be done by indicating $|E| - 2 \cdot |V| + 2$ affinely independent vertices of the polytope P(HAP). From the result for the assignment problem, we know that there are $|E_1| - 2 \cdot |V| + 2$ hyperassignments that are subsets of E_1 so that the corresponding polytope vertices are affinely independent. Thus, we have to indicate $|E| - |E_1| = |E \setminus E_1|$ more. This can be done as follows. Choose for every hyperedge e from $E \setminus E_1$ (which is, by construction, a proper hyperedge) some hyperassignment that consists only of e and edges from E_1. The corresponding vertices of the polytope and those we derived from the assignment problem are affinely independent since every variable for proper hyperedges is non-zero in exactly one of the vertices. □

6.2 An Extended Formulation

In this section, we will present an extended formulation of the canonical linear programming formulation (HAP) for the HAP. As we have discussed in Chapter 1, Formulation (HAP) is of a set partitioning type. Such a model can be strengthened by clique inequalities, which are usually difficult to separate.

Clique constraints are important for HAPs arising from real-world applications in railway vehicle rotation planning. Such instances are modeled using partitioned hypergraphs of maximum part size 7, and clique inequalities can significantly reduce the integrality gap of Model (HAP), see the computational results in [Heismann and Borndörfer, 2012].

It is already shown in the author's master's thesis [Heismann, 2010] that the extended formulation is of polynomial size, correct and implies all clique inequalities. Here, we will give an exact characterization of the inequalities implied by this extended formulation. The result about cliques can then be deduced as a corollary of this characterization.

Our extended integer linear programming formulation for the HAP is based on the notion of configurations introduced in Section 1.2. The configurations model the local incidence structure of hyperassignments at the parts of the hypergraph. The model is as follows.

$$\begin{aligned}
\underset{x\in\mathbb{R}^E, y\in\mathbb{R}^{\mathscr{C}_U}\times\mathbb{R}^{\mathscr{C}_V}}{\text{minimize}} \quad & \sum_{e\in E} c_E(e)x_e && && \text{(HAP_ext)}\\
\text{subject to} \quad & \sum_{e\in\delta(v)} x_e = 1 && \forall v\in U\cup V && \text{(i)}\\
& \sum_{C\in\mathscr{C}_U:\, e\in C} y_C = x_e && \forall e\in E && \text{(ii)}\\
& \sum_{C\in\mathscr{C}_V:\, e\in C} y_C = x_e && \forall e\in E && \text{(iii)}\\
& x, y \geq 0 && && \text{(iv)}\\
& x\in\mathbb{Z}^E && && \text{(v)}\\
& y\in\mathbb{Z}^{\mathscr{C}_U}\times\mathbb{Z}^{\mathscr{C}_V} && && \text{(vi)}
\end{aligned}$$

The model uses binary variables x_e and y_C for the choice of hyperedge e and configuration C, respectively. Constraints (HAP_ext) (i) are copied from the canonical formulation. Equations (HAP_ext) (ii) and (iii) link the hyperedges to the configurations in parts in U resp. V that contain them. (HAP_ext) (iv), (v), and (vi) enforce non-negativity and integrality.

This section resorts to the following notation. For an index set I, a subset $J\subseteq I$, and a vector $x\in\mathbb{R}^I$, denote by $x|J = x_J$ the projection of x onto the coordinates in J. Likewise, let $P|J$ denote the projection of a polytope $P\subseteq\mathbb{R}^I$ onto $\mathbb{R}^J$.

Let

$$P_{\text{LP}}(\text{HAP\_ext}) := \{(x,y)\in\mathbb{R}^E\times\mathbb{R}^{\mathscr{C}_U}\times\mathbb{R}^{\mathscr{C}_V} :\ \text{(HAP\_ext) (i)–(iv)}\}$$

be the polytope associated with the LP relaxation of the integer program (HAP_ext). Then $P_{\text{LP}}(\text{HAP\_ext})|E$ and $P_{\text{LP}}(\text{HAP\_ext})|\delta(\Pi)$ project the LP relaxation of the extended formulation onto the original space of all hyperedge variables and those incident to some part Π, respectively.

The following theorem relates the integer program (HAP_ext) to the formulation (HAP) and, in particular, proves the correctness of (HAP_ext).

Theorem 6.2.1 ([Heismann, 2010]). *Let $G=(U,V,E)$ be a partitioned hypergraph and $c_E : E\to\mathbb{R}$ a cost function. Then the projection*

$$\cdot|E : \mathbb{R}^E\times\mathbb{R}^{\mathscr{C}_U}\times\mathbb{R}^{\mathscr{C}_V},\ (x,y)\mapsto x$$

is a bijection between feasible solutions of (HAP_ext) *and* (HAP)*, and therefore hyperassignments in G. The optimum value of* (HAP_ext) *is equal to the cost of the minimum cost hyperassignment in G w. r. t. c_E if it exists and to ∞ otherwise.*

Proof. Let $x \in \mathbb{R}^E$ be the incidence vector of a hyperassignment H in G. We have to show that there is exactly one $y \in \mathbb{R}^{\mathscr{C}_U} \times \mathbb{R}^{\mathscr{C}_V}$ such that (x, y) is feasible for (HAP_ext). Define

$$\mathscr{C}'_U := \{\delta(\Pi) \cap H : \Pi \in \{U_1, \ldots, U_p\}\}, \mathscr{C}'_V := \{\delta(\Pi) \cap H : \Pi \in \{V_1, \ldots, V_q\}\}$$

as the set of intersections of the hyperassignment H with the hyperedges incident to the parts in U and V, respectively. Then $\mathscr{C}'_U$ and $\mathscr{C}'_V$ are two sets of configurations, namely, $\mathscr{C}'_U \subseteq \mathscr{C}_U$ and $\mathscr{C}'_V \subseteq \mathscr{C}_V$, and we can define the incidence vector $y \in \mathbb{R}^{\mathscr{C}_U} \times \mathbb{R}^{\mathscr{C}_V}$ of $\mathscr{C}'_U \cup \mathscr{C}'_V$.

We show next that the vector (x, y) is a solution of (HAP_ext). Equations (i) hold because x is the incidence vector of the hyperassignment H. Now consider some hyperedge e incident to the part Π, w. l. o. g. $\Pi \subseteq U$. Suppose $x_e = 1$, i. e., $e \in H$. Then e is contained in exactly one configuration in $\mathscr{C}'_U$, namely, $e \in H \cap \delta(\Pi)$, which means that Equation (HAP_ext) (ii) holds. If $x_e = 0$, i. e., $e \notin H$, then e is not contained in any configuration in $\mathscr{C}'_U$ and (HAP_ext) (ii) also holds. The case $\Pi \subseteq V$ and (HAP_ext) (iii) is analogous. Constraints (iv)–(vi) are clear.

To see that there is only one possible choice for y consider some part $\Pi \in U$ containing a node $v \in \Pi$. Substituting (HAP_ext) (ii) for the variables x_e in Constraint (HAP_ext) (i) associated with node v yields

$$\sum_{C \in \mathscr{C}_\Pi} y_C \overset{(*)}{=} \sum_{e \in \delta(v)} \sum_{C \in \mathscr{C}_U :\, e \in C} y_C = \sum_{e \in \delta(v)} x_e = 1;$$

i. e., y associates exactly one configuration with every part Π; Equation (*) holds because every configuration in $\mathscr{C}_\Pi$ contains exactly one hyperedge incident to node v. Equations (HAP_ext) (ii) and (iii) ensure that this configuration contains exactly the edges in $H \cap \delta(\Pi)$, i. e., the edges such that $x_e = 1$. This means that $y_C = 1$ exactly for $C = H \cap \delta(\Pi)$.

The cost statement is obvious. □

The next theorem describes the projection of the feasible solutions of the linear programming relaxation of the extended formulation to the hyperedge variables.

Theorem 6.2.2.

$$P_{LP}(HAP\_ext)|E = P_{LP}(HAP) \cap \Big\{x \in \mathbb{R}^E : \sum_{e\in E_1} \kappa_e x_e \leq \sum_{e\in E_2} \lambda_e x_e \\ \forall E_1, E_2 \subseteq E, \kappa \in \mathbb{Z}^{E_1}, \lambda \in \mathbb{Z}^{E_2} \text{ such that} \\ \sum_{e\in E_1:\ e\in C} \kappa_e \leq \sum_{e\in E_2:\ e\in C} \lambda_e \text{ for all } C \in \mathscr{C}_U \text{ or all } C \in \mathscr{C}_V\Big\},$$

i. e., the LP relaxation of the extended formulation (HAP_ext) *implies exactly the type of inequalities for the hyperedge variables, for which for each configuration for all for the parts on the U-side or all the parts on the V-side the following holds: For the hyperedges in the intersection of every two hyperedge subsets with the configuration, the sum of the coefficients on the smaller side of the inequality is less than or equal to the one for the greater side.*

Proof. **Direction "⊆".** As $P_{\text{LP}}(\text{HAP\_ext})|E \subseteq P_{\text{LP}}(\text{HAP})$, we have to show that for all $x \in P_{\text{LP}}(\text{HAP\_ext})$

$$\sum_{e\in E_1} \kappa_e x_e \leq \sum_{e\in E_2} \lambda_e x_e \tag{6.2}$$

if

$$\sum_{e\in E_1:\ e\in C} \kappa_e \leq \sum_{e\in E_2:\ e\in C} \lambda_e \tag{6.3}$$

for all $C \in \mathscr{C}_U$ or all $C \in \mathscr{C}_V$ to prove this direction. Let $E_1, E_2 \subseteq E$ be arbitrary but fixed hyperedge sets and $\kappa \in \mathbb{Z}^{E_1}$, $\lambda \in \mathbb{Z}^{E_2}$ that fulfill the requirement, i. e., Equation (6.3) holds for, say, all $C \in \mathscr{C}_U$. Observe that Equation (HAP_ext) (ii) for e has on the left-hand side coefficient 1 exactly for the y-variables associated with configurations $C \in \mathscr{C}_U$ which contain e and 0 for all the others. Sum up $\kappa_e, e \in E_1$ times Equations (HAP_ext) (ii) for $e \in E_1$. Also, sum up $\lambda_e, e \in E_2$ times Equations (HAP_ext) (ii) for $e \in E_2$. The last observation and the constraint on the choice of E_1 and E_2 imply that the coefficient of y_C for every $C \in \mathscr{C}_U$ on the left-hand side of the resulting equation for $e \in E_1$ is less than or equal to the one for $e \in E_2$. Since $y \geq 0$, the left-hand side of the resulting equation for $e \in E_1$ is also less than or equal to the left-hand side of the equation for $e \in E_2$, which of course then also holds for the right-hand sides of the equations. This is exactly Inequality (6.2), which completes Direction "⊆" of the proof.

Direction "⊇". We will employ Fourier-Motzkin elimination for this second direction of the proof. Since all the equations and inequalities describing $P_{\text{LP}}(\text{HAP})$ are exactly those in (HAP_ext) that do not contain any y-variables, the theorem says that Fourier-Motzkin elimination of all y-variables in the LP

relaxation of (HAP_ext) leads exactly to Inequalities (6.2) for all $E_1, E_2 \subseteq E$, $\kappa \in \mathbb{Z}^{E_1}, \lambda \in \mathbb{Z}^{E_2}$ such that the requirement (6.3) holds for all $C \in \mathscr{C}_U$ or all $C \in \mathscr{C}_V$. By Direction "$\subseteq$" of the proof, all these inequalities are at least implied by the result of the Fourier-Motzkin elimination. Thus, we have to prove that inequalities resulting from the Fourier-Motzkin elimination cannot have other types than this. This will be shown now by induction. Every variable y_C appears either in the Equations (HAP_ext) (ii) or (HAP_ext) (iii), and in the non-negativity constraints (HAP_ext) (iv). We will w. l. o. g. deal only with y_C for $C \in \mathscr{C}_U$ and therefore only Constraints (HAP_ext) (ii) and (iv).

More specifically, our induction hypothesis is that for every inequality (and also equation—but it can be viewed as a combination of the corresponding "$\leq$" and "$\geq$" inequalities) throughout the Fourier-Motzkin elimination after every step, i. e., elimination of some variable, the following three invariants hold.

I1 There exist sets of hyperedges $E_1, E_2 \subseteq E$ such that the inequalities can be written as

$$\sum_{e \in E_1} \kappa_e x_e - \sum_{C \in \mathscr{C}_U} \mu_C y_C \leq \sum_{e \in E_2} \lambda_e x_e$$

for some $\kappa \in \mathbb{Z}^{E_1}, \lambda \in \mathbb{Z}^{E_2}, \mu \in \mathbb{Z}^{\mathscr{C}_U}$.

I2 For every $C \in \mathscr{C}_U$ that has not been eliminated yet,

$$-\mu_C \leq \sum_{e \in E_2 \colon e \in C} \lambda_e - \sum_{e \in E_1 \colon e \in C} \kappa_e.$$

(For the others, the coefficient has to be 0.)

I3 For every $C \in \mathscr{C}_U$ that has been eliminated already, Inequality (6.3) holds.

They obviously imply that the result after Fourier-Motzkin elimination of all y-variables is as stated above.

Basis: For all Equations (HAP_ext) (ii) *and Inequalities* (iv)*—this is the input for the Fourier-Motzkin elimination—the claims I1–I3 hold.* For (HAP_ext) (ii), the correctness of Claims I1 and I2 can be seen by choosing $E_1 = \emptyset$, $E_2 = \{e\}$, $\lambda_e = \pm 1$, and $\mu_C = \mp 1$ if $e \in C$, 0 otherwise. For Constraints (HAP_ext) (iv), choose $E_1 = E_2 = \emptyset$ and $\mu_C = 1$ for one configuration $C \in C_U$, $\mu_C = 0$ for the others. Claim I3 is true as no $C \in \mathscr{C}_U$ has been eliminated already.

Inductive step: If for all inequalities after some Fourier-Motzkin elimination step Claims I1–I3 hold, then this is also true after the elimination of another variable $y_{C'}$ that has not been eliminated yet for some $C' \in \mathscr{C}_U$. We perform Fourier-Motzkin elimination for the variable $y_{C'}$: Take two inequalities

$$\sum_{e \in E_1} \kappa_e x_e - \sum_{C \in \mathscr{C}_U} \mu_C y_C \leq \sum_{e \in E_2} \lambda_e x_e$$

and

$$\sum_{e\in E_1'} \kappa_e' x_e - \sum_{C\in\mathscr{C}_U} \mu_C' y_C \leq \sum_{e\in E_2'} \lambda_e' x_e$$

where $\mu_{C'}, \mu_{C'}' \neq 0$, and solve them for $y_{C'}$. If $\mu_{C'}$ and $\mu_{C'}'$ are greater than 0, this results in

$$y_{C'} \geq \sum_{e\in E_1} \frac{\kappa_e}{\mu_{C'}} x_e - \sum_{e\in E_2} \frac{\lambda_e}{\mu_{C'}} x_e - \sum_{C\in\mathscr{C}_U:\ C\neq C'} \frac{\mu_C}{\mu_{C'}} y_C$$

and

$$y_{C'} \geq \sum_{e\in E_1'} \frac{\kappa_e'}{\mu_{C'}'} x_e - \sum_{e\in E_2'} \frac{\lambda_e'}{\mu_{C'}'} x_e - \sum_{C\in\mathscr{C}_U:\ C\neq C'} \frac{\mu_C'}{\mu_{C'}'} y_C.$$

Otherwise the $\geq$ sign has to be changed into a $\leq$ sign in the corresponding inequality.

This implies that we have to combine the the two inequalities in the Fourier-Motzkin elimination if and only if $\mu_{C'}$ and $\mu_{C'}'$ have different algebraic signs. W. l. o. g., assume that $\mu_{C'} < 0$ and $\mu_{C'}' > 0$. This yields

$$\sum_{e\in E_1} \frac{\kappa_e}{\mu_{C'}} x_e - \sum_{e\in E_2} \frac{\lambda_e}{\mu_{C'}} x_e - \sum_{C\in\mathscr{C}_U, C\neq C'} \frac{\mu_C}{\mu_{C'}} y_C \geq \sum_{e\in E_1'} \frac{\kappa_e'}{\mu_{C'}'} x_e - \sum_{e\in E_2'} \frac{\lambda_e'}{\mu_{C'}'} x_e - \sum_{C\in\mathscr{C}_U, C\neq C'} \frac{\mu_C'}{\mu_{C'}'} y_C.$$

By reordering we get

$$\sum_{e\in E_1} \frac{\kappa_e}{\mu_{C'}} x_e + \sum_{e\in E_2'} \frac{\lambda_e'}{\mu_{C'}'} x_e - \sum_{C\in\mathscr{C}_U, C\neq C'} \frac{\mu_C}{\mu_{C'}} y_C - \sum_{C\in\mathscr{C}_U, C\neq C'} \left(-\frac{\mu_C'}{\mu_{C'}'}\right) y_C \geq \sum_{e\in E_1'} \frac{\kappa_e'}{\mu_{C'}'} x_e + \sum_{e\in E_2} \frac{\lambda_e}{\mu_{C'}} x_e.$$

Then, multiplication with $\mu_{C'}\mu_{C'}'$ yields

$$\sum_{e\in E_1} \kappa_e \mu_{C'}' x_e + \sum_{e\in E_2'} \lambda_e' \mu_{C'} x_e - \sum_{C\in\mathscr{C}_U, C\neq C'} \mu_C \mu_{C'}' y_C - \sum_{C\in\mathscr{C}_U, C\neq C'} \left(-\mu_C' \mu_{C'}\right) y_C \leq \sum_{e\in E_1'} \kappa_e' \mu_{C'} x_e + \sum_{e\in E_2} \lambda_e \mu_{C'}' x_e.$$

For the last inequality, Claim I1 is true. The two hyperedge sets are $E_1 \cup E_2'$ and $E_1' \cup E_2$.

For Claim I2, for every $C \in \mathscr{C}_U$ that has not been eliminated yet,

$$\begin{aligned}&-\mu'_{C'}\mu_C + \mu_{C'}\mu'_C \\ &\qquad \leq -\mu'_{C'}\left(\sum_{e\in E_1:\ e\in C}\kappa_e - \sum_{e\in E_2:\ e\in C}\lambda_e\right) + \mu_{C'}\left(\sum_{e\in E_1':\ e\in C}\kappa'_e - \sum_{e\in E_2':\ e\in C}\lambda'_e\right)\end{aligned}$$

has to hold. This is true because Inequality (6.3) holds for the two initial inequalities, to which we applied the elimination step, by induction hypothesis, and $\mu'_{C'} > 0$, $\mu_{C'} < 0$.

Claim I3 holds for all configurations $C \in \mathscr{C}_U$ that have been eliminated before C' by induction hypothesis and adding the required inequalities in Claim I3 for the two initial inequalities. For C', Claim I3 for the resulting inequality means that

$$\sum_{e\in E_1:\ e\in C'}\kappa_e\mu'_{C'} + \sum_{e\in E_2':\ e\in C'}\lambda'_e\mu_{C'} \leq \sum_{e\in E_1':\ e\in C'}\kappa'_e\mu_{C'} + \sum_{e\in E_2:\ e\in C'}\lambda_e\mu'_{C'}$$

has to hold. Reordering yields

$$\begin{aligned}&-\mu_{C'}\left(\sum_{e\in E_2':\ e\in C'}\lambda'_e - \sum_{e\in E_1':\ e\in C'}\kappa'_e\right) \\ &\qquad\qquad \geq -\mu'_{C'}\left(\sum_{e\in E_2:\ e\in C'}\lambda_e - \sum_{e\in E_1:\ e\in C'}\kappa_e\right).\end{aligned}$$

This is true because of Claim I2 for the induction hypothesis and $\mu'_{C'} > 0$, $\mu_{C'} < 0$. □

One class of inequalities implied in this way are the clique inequalities. To show this, we will need the following lemma.

Lemma 6.2.3 ([Heismann, 2010])**.** *Let $G = (U, V, E)$ be a partitioned hypergraph and Q a clique in G. Then there exists a part Π such that $Q \subseteq \delta(\Pi)$, i. e., every clique is a subset of the set of hyperedges incident to some part Π in G.*

Proof. Let Q be a nonempty clique in G (otherwise the lemma is trivial) containing some hyperedge e_1. Let $e_1 \cap U$ be contained in some part U_1 and $e_1 \cap V$

be contained in some part V_1. Every other hyperedge e in Q must either satisfy $e \cap U \subseteq U_1$ or $e \cap V \subseteq V_1$, otherwise e_1 and e would have an empty intersection. Assume the statement does not hold and Q contains some hyperedge e_2 such that $e_2 \cap U \nsubseteq U_1$ and some hyperedge e_3 such that $e_3 \cap V \nsubseteq V_1$. Then $e_2 \cap e_3$ must be empty since both $e_2 \cap U$, $e_3 \cap U$ and $e_2 \cap V$, $e_3 \cap V$ are contained in different parts. This contradicts the assumption that Q is a clique. Hence, either U_1 or V_1 contains $e \cap U$ or $e \cap V$ for all hyperedges $e \in Q$. □

Corollary 6.2.4. *Let $G = (U, V, E)$ be a partitioned hypergraph and $Q \subseteq E$ be a clique. Then, the clique inequality*

$$\sum_{e \in Q} x_e \leq 1$$

is satisfied by all feasible solutions of the LP relaxation of (HAP_ext).

Proof. By Lemma 6.2.3, $Q \subseteq \delta(\Pi)$ for some part Π. W. l. o. g., let $\Pi \subseteq U$ (the formulation is symmetric in U, V). Further, let v be some vertex in Π.

For

$$\begin{aligned} E_1 &:= Q, \\ E_2 &:= \delta(v), \\ \kappa_e &:= 1 \text{ for all } e \in Q, \\ \lambda_e &:= 1 \text{ for all } e \in \delta(v), \end{aligned}$$

the condition from Theorem 6.2.2 reads:

$$\sum_{e \in Q:\ e \in C} \kappa_e \leq \sum_{e \in \delta(v):\ e \in C} \lambda_e \text{ for all } C \in \mathscr{C}_U.$$

For $C \in \mathscr{C}_U$ that are not in $\mathscr{C}_\Pi$, this is true since $E_1 \cap C = \delta(v) \cap C = \emptyset$. For all $C \in \mathscr{C}_\Pi$ the condition is also fulfilled since

$$\begin{aligned} \sum_{e \in Q:\ e \in C} \kappa_e &= |Q \cap C| \\ &\leq 1 \\ &= |\delta(v) \cap C| \\ &= \sum_{e \in \delta(v):\ e \in C} \lambda_e. \end{aligned}$$

The $\leq$ sign holds because the hyperedges in a configuration are pairwise disjoint and those in Q are not. The equality sign thereafter holds since by definition of

a configuration in a part Π there has to be exactly one hyperedge in it incident to this vertex.

Therefore, by Theorem 6.2.2

$$\begin{aligned}\sum_{e\in Q} x_e &= \sum_{e\in Q} \kappa_e x_e \\ &\leq \sum_{e\in\delta(v)} \lambda_e x_e \\ &\leq \sum_{e\in\delta(v)} x_e \\ &= 1\end{aligned}$$

holds for all $x \in P_{\mathrm{LP}}(\mathrm{HAP\_ext})$. □

6.3 A Generalization of Odd Set Inequalities for the Set Packing Problem

The set packing problem is well-known in combinatorial optimization and has a wide range of applications. Although many classes of facets for the set packing polytope $P(\mathrm{SSP})$ are known (see, e. g., [Borndörfer, 1998]), there is still no complete polyhedral description and further facet classes have to be researched. In our analysis of the HAP polytope, we found inequalities that are valid not only for the HAP but for set packing polytopes in general. 13 of the otherwise not interpreted facet classes of the HAP polytope for the hypergraph $G_{2,3}$ (see Section 5.5) are of this type.

A polynomially solvable special case of the set packing problem, in which all sets in E have size two, is the matching problem. For this problem, the polytope can be completely described by adding so-called odd set inequalities to the canonical description [Edmonds, 1965a]. In this section, we show how, employing cliques, the odd set inequalities can be generalized to valid inequalities for the set packing problem polytope with a clear combinatorial meaning. We also relate the presented inequality class to a different generalization of odd set inequalities for the stable set problem called general clique family inequalities [Pêcher and Wagler, 2006].

We first present a combinatorial derivation of the inequality class in Subsection 6.3.1. Then, we show a comparison with the general clique family inequalities in Subsection 6.3.2, and how both inequality classes can be combined in Subsection 6.3.3. In the end, we will state IPs that can be used to check whether

a given facet inequality has this type in Subsection 6.3.4, and also a separation IP in Subsection 6.3.5.

6.3.1 Generalizing Odd Set Inequalities

Consider the set packing problem for the hypergraph $G = (V, E)$.

In the special case that G is a graph, the set packing problem becomes a matching problem which can be completely described by the following system of inequalities [Edmonds, 1965a]:

$$\sum_{e \in \delta(v)} x_e \leq 1 \qquad \forall v \in V \qquad \text{(MP) (i)}$$

$$\sum_{e \in E:\ e \subset V'} x_e \leq \frac{|V'|-1}{2} \quad \forall V' \subseteq V,\ |V'| \geq 3,\ |V'| \text{ odd} \qquad \text{(MP) (ii)}$$

$$x \geq 0 \qquad \forall e \in E \qquad \text{(MP) (iii)}$$

Inequalities (MP) (ii) are called odd set inequalities. Their combinatorial meaning is that for every odd set $V' \subseteq V$ of $|V'| = 2k+1$ vertices there can be at most $\left\lfloor \frac{|V'|}{2} \right\rfloor = \frac{|V'|-1}{2} = k$ edges connecting pairs of them in a matching. This holds since every edge is incident to two vertices in k, every vertex can be incident to at most one edge in a matching, and $k+1$ edges would need therefore already $2k+2 > |V'|$ distinct vertices. For an example, see Figure 6.1.

A formal proof of validity for odd set inequalities can be interpreted as a Chvátal-Gomory procedure with coefficient $\frac{1}{2}$ for all inequalities of type (SSP) (i) for $v \in V'$ and 0 for all others.

We will generalize these inequalities for the set packing problem, i. e., from graphs to hypergraphs, in three steps. Step 1 will adapt the odd set inequalities to p-uniform hypergraphs, i. e., to hypergraphs which have hyperedges all of size p, where p can be greater than two. Then, we will tackle hypergraphs with hyperedges of arbitrary size by viewing them as combinations of hyperedges of size p in Step 2. Step 3 will generalize sets of hyperedges incident to one vertex to cliques.

Odd set inequalities can be written also as

$$\sum_{e \in E} \left\lfloor \frac{|\{v \in V' : e \in \delta(v)\}|}{2} \right\rfloor x_e \leq \left\lfloor \frac{|V'|}{2} \right\rfloor,$$

which is a more useful representation for our generalization procedure.

Step 1. Let G be p-uniform. Applying the idea of odd set inequalities to this situation yields that for every set $V' \subseteq V$ of $|V'| = pk+r$, $0 \leq r \leq p-1$, $k, r \in \mathbb{N}$

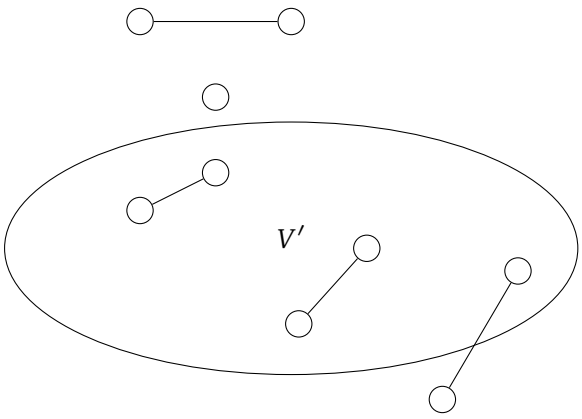


Figure 6.1: A matching in a graph $G = (V, E)$ with nine vertices and a vertex subset V' surrounded by an ellipse. There can be only $k = 2$ edges connecting the $2k + 1 = 5$ vertices in V'. All other edges in the matching have to contain at least one vertex from $V \setminus V'$.

vertices there can be at most $\left\lfloor \frac{|V'|}{p} \right\rfloor = \frac{|V'|-r}{p} = k$ hyperedges, each connecting p of them, in a packing. For an example, see Figure 6.2.

This leads to the inequality

$$\sum_{e \in E} \left\lfloor \frac{|\{v \in V' : e \in \delta(v)\}|}{p} \right\rfloor x_e \leq \left\lfloor \frac{|V'|}{p} \right\rfloor \quad \forall V' \subseteq V.$$

The coefficients $\left\lfloor \frac{|\{v \in V' : e \in \delta(v)\}|}{p} \right\rfloor$ all have value 0 or 1.

The inequality can be also derived using a Chvátal-Gomory procedure with coefficient $\frac{1}{p}$ for all inequalities of type (SSP) (i) for $v \in V'$ and 0 for all others.

Step 2. Let G be an arbitrary hypergraph. Choose some $p \in \mathbb{N}$, $p \geq 2$. Contrary to the previous case, where all hyperedges had size p, there now might be hyperedges in the packing that contain more than p vertices from V'. The inequality from Step 1, however, is still true. A hyperedge that contains $kp + r$, $0 \leq r \leq p - 1$, $k, r \in \mathbb{N}$ vertices from V' can be viewed as k hyperedges of size p that are contained in V'. For an example, see Figure 6.3.

This idea leads to the same inequality class as in Step 1

$$\sum_{e \in E} \left\lfloor \frac{|\{v \in V' : e \in \delta(v)\}|}{p} \right\rfloor x_e \leq \left\lfloor \frac{|V'|}{p} \right\rfloor \quad \forall V' \subseteq V$$

for arbitrary hypergraphs. The coefficients $\left\lfloor \frac{|\{v \in V' : e \in \delta(v)\}|}{p} \right\rfloor$ may now have a value greater than 1.

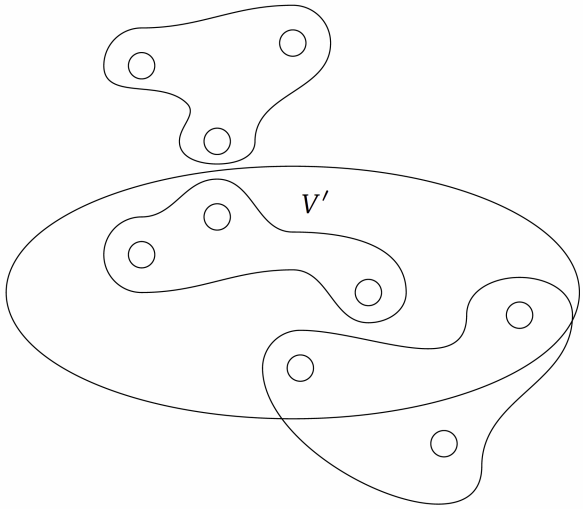


Figure 6.2: A packing in a 3-uniform hypergraph $G = (V, E)$ with nine vertices and a vertex subset V' (surrounded by an ellipse). There can be at most $\lfloor \frac{5}{3} \rfloor = 1$ hyperedge which is a subset of the five vertices in V'. All other hyperedges in the packing have to have at least one vertex from $V \setminus V'$.

As in the last step, a Chvátal-Gomory procedure with coefficient $\frac{1}{p}$ for all inequalities of type (SSP) (i) for $v \in V'$ and 0 for all others yields these inequalities.

The idea from Step 2 can also be applied to other inequalities. In general, we can prove the following theorem.

Theorem 6.3.1. *Let $G = (V, E)$ be a hypergraph. Let $G' = (V, E')$ be another hypergraph with the same vertex set such that for every $e \in E$ there exists a set $R_e \subseteq E'$ which is a partition of e, i. e., all $e' \in R_e$ are pairwise disjoint and $e = \bigcup_{e' \in R_e} e'$. Further, let*

$$\sum_{e' \in E'} a_{e'} x_{e'} \leq b$$

be a valid inequality for the set packing polytope $P(SSP)$ for G'. Then

$$\sum_{e \in E} \left(\sum_{e' \in R_e} a_{e'} \right) x_e \leq b$$

is a valid inequality for the set packing polytope $P(SSP)$ for G.

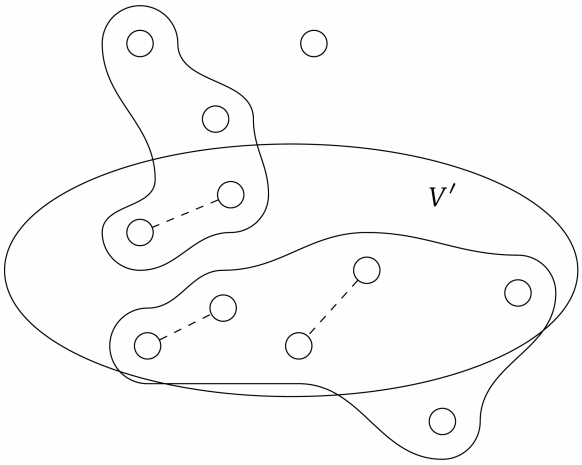


Figure 6.3: A packing in a hypergraph $G = (V, E)$ with eleven vertices and a vertex subset V' (surrounded by an ellipse), $p = 2$. There can be at most $\lfloor \frac{7}{p} \rfloor = 3$ pairwise disjoint edges which are subsets of both the seven vertices in V' and some hyperedge in the packing. The edges are indicated by dashed lines in the visualization. All other possible edges that would connect two vertices that are contained in some hyperedge in the packing have to have at least one vertex from $V \setminus V'$.

Proof. It suffices to prove that

$$\sum_{e\in E}\left(\sum_{e'\in R_e} a_{e'}\right)x_e \leq b$$

holds for the vertices of the set packing polytope $P(\text{SSP})$ for G, i. e., for all the characteristic vectors $\chi(H) \in \mathbb{R}^E$ with

$$\chi_e(H) = \begin{cases} 1 & \text{if } e \in H \\ 0 & \text{otherwise.} \end{cases}$$

for packings H in G. For a packing H in G, $H' = \bigcup_{e\in H} R_e$ is a packing in G' since, by construction of the sets R_e, H' is a set of pairwise disjoint hyperedges $e' \in E'$. Thus, for the characteristic vector $\chi(H') \in \mathbb{R}^{E'}$ of H' with

$$\chi_{e'}(H') = \begin{cases} 1 & \text{if } e' \in H' \\ 0 & \text{otherwise} \end{cases}$$

the inequality

$$\sum_{e'\in E'} a_{e'}\chi_{e'}(H') \leq b$$

holds. Since the summand $a_{e'}\chi_{e'}(H')$ is 0 anyway if $e' \notin H'$

$$\sum_{e'\in H'} a_{e'}\chi_{e'}(H') \leq b$$

also holds. By construction $\chi_{e'_1}(H') = \chi_{e'_2}(H') = \chi_e(H)$ for all $e'_1, e'_2 \in R_e$ for every $e \in H$, so the last inequality can also be written as

$$\sum_{e\in H}\left(\sum_{e'\in R_e} a_{e'}\right)\chi_e(H) \leq b.$$

Since $\chi_e(H)$ is equal to 0 for all $e \notin H$ the following inequality can be obtained from the previous one by adding zeros to the left-hand side and is therefore also true:

$$\sum_{e\in E}\left(\sum_{e'\in R_e} a_{e'}\right)\chi_e(H) \leq b.$$

This proves the theorem. □

Step 3. For the third step, observe that for every vertex v in a graph or hypergraph, $\delta(v)$ is a clique. To get the odd set inequalities or their generalizations in Steps 1 and 2, the Chvátal-Gomory procedure could be applied to the inequalities of type (SSP) (i), which are clique inequalities. In a graph, $\delta(v)$ is the only type of maximal edge cliques. However, there may be other cliques and therefore also other valid clique inequalities for a hypergraph. Applying the previous ideas to also other types of cliques for some clique set $\mathcal{Q}' \subseteq \mathcal{Q}$ we get the the inequalities

$$\sum_{e \in E} \left\lfloor \frac{|\{Q \in \mathcal{Q}' : \ e \in Q\}|}{p} \right\rfloor x_e \leq \left\lfloor \frac{|\mathcal{Q}'|}{p} \right\rfloor \quad \forall \mathcal{Q}' \subseteq \mathcal{Q}, p \in \mathbb{N}, p \geq 2. \quad \text{(OSI\_SSP)}$$

Please note that each inequality for some selection of cliques $\mathcal{Q}'$ is implied by some inequality for a selection of maximal cliques $\mathcal{Q}''$. This holds since there exists for each set $\mathcal{Q}'$ a set $\mathcal{Q}''$ that contains for each clique in $\mathcal{Q}'$ some maximal clique containing it. All the coefficients in Inequality (OSI_SSP) for $\mathcal{Q}''$ will be equal or to or greater than the ones for $\mathcal{Q}'$.

For the HAP, these inequalities can be facet-defining. In the HAP polytope for the hypergraph $G_{2,3}$, 13 of the 30 facet classes (these 30 equivalence classes w. r. t. symmetry contain all together 14049 facets) that are otherwise not interpreted can be described in this way with $p = 2$. Also the three other facet classes that we have understood, the clique inequalities and the two types of non-negativity constraints, can be viewed as such inequalities if we allow $\mathcal{Q}'$ to be a multiset and contain some cliques multiple times.

6.3.2 Comparison with General Clique Family Inequalities

[Pêcher and Wagler, 2006] propose a different generalization of odd set cuts for the set packing problem. These inequalities, which they call "general clique family inequalities", have a similar structure (division by some $p \in \mathbb{N}$, rounding, coefficient for a hyperedge variable depends on the number of cliques that contain this hyperedge), however, the resulting inequality is different. Also, to the best of our knowledge, no combinatorial interpretation was developed for general clique family inequalities so far.

General clique family inequalities are defined as follows. Let $\mathcal{Q}' \subseteq \mathcal{Q}$ be a set of at least three cliques for the hypergraph $G = (V, E)$. Choose an integer p with $2 \leq p \leq |\mathcal{Q}'|$, define $R := |\mathcal{Q}'| \mod p$ and choose an integer J with $0 \leq J \leq p - R$. Now define $E_i := \left\{e \in E : \ |\{Q \in \mathcal{Q}' : \ e \in Q\}| = i\right\}$ for $i \in \{1, 2, \ldots, |\mathcal{Q}'|\}$ to be the set of hyperedges that are contained in exactly i cliques in $\mathcal{Q}'$. The

general clique family inequality

$$\sum_{i=p}^{|\mathcal{Q}'|}(p-R)\sum_{e\in E_i}x_e+\sum_{j=1}^{J}(p-R-j)\sum_{e\in E_{p-j}}x_e\leq b$$

is valid if $b\geq(p-R)\left\lfloor\frac{|\mathcal{Q}'|}{p}\right\rfloor$.

To compare the general clique family inequalities to our Inequalities (OSI_SSP) from Step 3 in Subsection 6.3.1, we rewrite them as

$$\sum_{i=0}^{|\mathcal{Q}'|}\left\lfloor\frac{i}{p}\right\rfloor\sum_{e\in E_i}x_e\leq\left\lfloor\frac{|\mathcal{Q}'|}{p}\right\rfloor. \tag{OSI_SSP$'$}$$

Then, we divide both sides of the general clique family inequalities with strongest allowed b by $(p-R)$ to get the valid inequality

$$\sum_{i=p}^{|\mathcal{Q}'|}\sum_{e\in E_i}x_e+\sum_{j=1}^{J}\frac{p-R-j}{p-R}\sum_{e\in E_{p-j}}x_e\leq\left\lfloor\frac{|\mathcal{Q}'|}{p}\right\rfloor. \tag{GCFI}$$

Now the right-hand sides are equal. The coefficients of x_e, $e\in E$ on the left-hand sides are summarized in Table 6.1 depending on the number $i:=|\{Q\in\mathcal{Q}':\ e\in Q\}|$ of edge cliques in $\mathcal{Q}'$ that contain e. The table shows that the inequalities concentrate on coefficients for different kinds of hyperedge variables although they employ similar objects. The inequalities derived in this section have non-zero coefficients only for hyperedges of size $\geq p$. These coefficients may differ depending on the hyperedge size and be >1, whereas the corresponding coefficients in the general clique family inequalities are all equal to 1. General clique family inequalities, however, have non-zero coefficients for smaller hyperedges.

Thus, the inequality class presented in this section is different from the general clique family inequalities.

6.3.3 Strengthening General Clique Family Inequalities

The following theorem gives rise to a strengthening opportunity for general clique family inequalities. It shows a similar idea as Theorem 6.3.1 for cliques instead of vertices but relies on stronger assumptions: The inequality that is used to generate a new one has to be valid for *every* set packing instance. This requirement is utilized to overcome the challenge to describe hypergraphs with "divided clique memberships" for hyperedges, which seems to be not as easy as this is with "vertex memberships".

Table 6.1: Coefficient of x_e, $e \in E$ on left-hand sides of Inequalities (OSI_SSP′) derived in Step 3 and General Clique Family Inequalities (GCFI) depending on the number $i := |\{Q \in \mathcal{Q}' : e \in Q\}|$ of cliques in $\mathcal{Q}'$ that contain e.

	(OSI_SSP′)	(GCFI)
$0 \leq i < p - J$	0	0
$p - J \leq i < p$	0	$\frac{i-R}{p-R}$
$p \leq i \leq \lvert\mathcal{Q}'\rvert$	$\left\lfloor \frac{i}{p} \right\rfloor$	1

Theorem 6.3.2. *Let*

$$\sum_{i=1}^{|\mathcal{Q}'|} a_i \sum_{e \in E_i} x_e \leq b$$

be an inequality which is valid for the set packing polytope of every hypergraph $G = (V, E)$ and every set $\mathcal{Q}' \subseteq \mathcal{Q}$ of cliques in G with

$$E_i := \left\{ e \in E : |\{Q \in \mathcal{Q}' : e \in Q\}| = i \right\} \text{ for } i \in \{1, 2, \ldots, |\mathcal{Q}'|\}.$$

For every $i \in \{1, 2, \ldots, |\mathcal{Q}'|\}$, let K_i be a multiset of natural numbers such that $\sum_{k \in K_i} k = i$. Define $a'_i = \sum_{k \in K_i} a_k$. Then, the inequality

$$\sum_{i=1}^{|\mathcal{Q}'|} a'_i \sum_{e \in E_i} x_e \leq b \tag{6.5}$$

is also valid for the set packing polytope of every hypergraph $G = (V, E)$.

Proof. As in the proof of Theorem 6.3.1, we will work with the characteristic vectors. However, we will empoy the stable set representation of the set packing problem, i. e., the conflict graph, and will have to do a construction to get the instance to which we will apply an inequality that is assumed to be valid in this theorem. Let $G = (V, E)$ be a hypergraph, $\mathcal{Q}'$ a set of cliques in G, and $H \subseteq E$ a hyperassignment in G. We have to show that the characteristic vector $\chi(H) \in \{0, 1\}^E$ of the hyperassignment fulfills Inequality (6.5). Using an inductive argument, it suffices to show this for a setting where $|K_j| = 2$ for one $j \in \{1, 2, \ldots, |\mathcal{Q}'|\}$ and $|K_j| = 1$ for all the others, i. e., $a'_i \neq a_i$ only for $i = j$. Repeated application of this case implies all other cases.

Denote the two elements of K_j by j' and j''. For every hyperedge $e \in E_j$ arbitrarily partition the set of j cliques from $\mathcal{Q}'$ in which it is contained into two sets $\mathcal{Q}'_{e'}$ and $\mathcal{Q}'_{e''}$ of j' and j'' cliques, respectively.

We construct from the conflict graph $\text{conf}(G) = (V_{\text{conf}}, E_{\text{conf}})$ of G with $V_{\text{conf}} = E$ a new conflict graph $\text{conf}(G') = (V'_{\text{conf}}, E'_{\text{conf}})$ in the following way: Let

$$
\begin{aligned}
V'_{\text{conf}} &:= E \setminus E_j \cup \{e', e'' : e \in E_j\},\\
E'_{\text{conf}} &:= E'_{1\text{conf}} \cup E'_{2\text{conf}} \cup E'_{3\text{conf}} \cup E'_{4\text{conf}},\\
E'_{1\text{conf}} &:= \{\{f, g\} \in E_{\text{conf}} : E_j \cap \{f, g\} = \emptyset\},\\
E'_{2\text{conf}} &:= \Big\{\{e', f\}, \{e'', f\} : e \in E_j, \{e, f\} \in E_{\text{conf}},\\
&\qquad\qquad f \in V'_{\text{conf}} \setminus \{g', g'' : \ g \in \bigcup (\mathcal{Q}'_{e'} \cup \mathcal{Q}'_{e''})\}\Big\},\\
E'_{3\text{conf}} &:= \Big\{\{e', f\} : e \in E_j, f \in \bigcup \mathcal{Q}'_{e'}\Big\},\\
E'_{4\text{conf}} &:= \Big\{\{e'', f\} : e \in E_j, f \in \bigcup \mathcal{Q}'_{e''}\Big\}.
\end{aligned}
$$

The construction step doubles every $e \in E_j$. All conflict graph edges that are not connected to some $e \in E_j$ remain $\left(E'_{1\text{conf}}\right)$. All conflict graph edges that connected some $e \in E_j$ with some $f \in E$ such that e and f are not both contained in some clique in $\mathcal{Q}'$ are transformed to two edges connecting e' and e'' with f $\left(E'_{2\text{conf}}\right)$. The remaining conflict graph edges—they contain some $e \in E_j$ and another hyperedge f involved in a clique that contains e—are transformed to one or two edges that connects e' or/and e'' with f depending on cliques from which sets $\mathcal{Q}'_{e'}$ and $\mathcal{Q}'_{e''}$ f is contained in $\left(E'_{3\text{conf}}\right.$ and $\left.E'_{4\text{conf}}\right)$. For an example of this construction, see Figure 6.4.

Every stable set H in $\text{conf}(G)$ gives rise to a stable set

$$H' := \{e \in E \setminus E_j : \ e \in H\} \cup \{e', e'' \in E_j : \ e \in H\}$$

in $\text{conf}(G')$. To see that H' really is a stable set, assume that for some $f, g \in H'$ $\{f, g\} \in E'_{\text{conf}}$ and distinguish the following cases.

Case 1: $f, g \in E \setminus E_j$**.** Then $f, g \in H$ and $\{f, g\} \in E_{\text{conf}}$. Thus H is not a stable set in $\text{conf}(G)$. This contradicts the assumption.

Case 2: $f = e'$ **or** $f = e''$ **for some** $e \in E_j, g \in E \setminus E_j$**.** Then $e, g \in H$ and $\{e, g\} \in E_{\text{conf}}$. Thus, again, H is not a stable set in $\text{conf}(G)$, and this contradicts the assumption.

Case 3: $f = d'$ **or** $f = d''$**,** $g = e'$ **or** $f = e''$ **for some** $d, e \in E_j$**.** Then $d, e \in H$ and $\{d, e\} \in E_{\text{conf}}$. As above, this is a contradiction.

Case 4: $f = e'$ **and** $g = e''$ **for some** $e \in E_j$**.** This is not possible since by construction $\{e', e''\} \notin E'_{\text{conf}}$.

Now, an easy calculation is left to complete the proof. Let

$$\chi(H') \in \{0,1\}^{E\setminus E_j \cup \{e',e'':e\in E_j\}}$$

be the characteristic vector of H'. We know that

$$\sum_{i=1}^{|\mathcal{Q}'|} a_i \sum_{e\in E_i} \chi_e(H') \le b.$$

By construction $\chi_e(H') = \chi_e(H)$ for $e \notin E_j$, and there are no hyperedges in H' that are contained in exactly j cliques from $\mathcal{Q}'$ (these hyperedges from E were substituted in V'_{conf} by hyperedges that are now contained in j' or j'' cliques). Thus, the last inequality is equivalent to

$$\sum_{i=1,i\neq j}^{|\mathcal{Q}'|} a_i \sum_{e\in E_i} \chi_e(H) + a_{j'} \sum_{e\in E_j} \chi_{e'}(H') + a_{j''} \sum_{e\in E_j} \chi_{e''}(H') \le b.$$

The construction is such that $\chi_e(H) = \chi_{e'}(H') = \chi_{e''}(H')$ for all $e \in E_j$. Thus the last inequality implies

$$\sum_{i=1,i\neq j}^{|\mathcal{Q}'|} a_i \sum_{e\in E_i} \chi_e(H) + (a_{j'} + a_{j''}) \sum_{e\in E_j} \chi_e(H) \le b,$$

which was the proposition we had to prove. □

The last theorem can be applied to general clique family inequalities. To check how this could lead to the best possible strengthening of the inequality, i. e.,

$$a'_i = \max\left\{ \sum_{k\in K_i} a_k : \; K_i \text{ multiset of natural numbers, } \sum_{k\in K_i} k = i \right\} > a_i,$$

we distinguish the following cases (for an overview of the values of a_i, see the last column of Table 6.1):

Case 1: $0 \le i < p - J$**.** Here, all a_i are 0 and all a_j for $j < i$ are also 0. Thus, $a'_i = \sum_{k\in K_i} a_k = 0$ for all possible choices of K_i since $k \le i$ for all $k \in K_i$ and a_i cannot be increased.

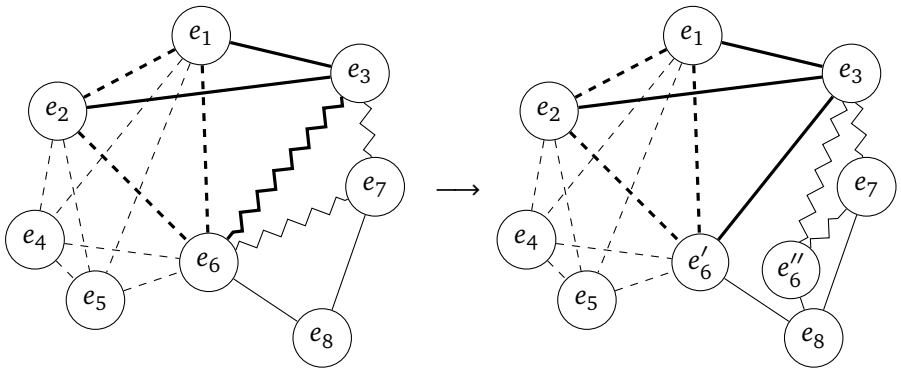


Figure 6.4: A conflict graph with three cliques in the set $\mathcal{Q}' = \big\{\{e_1, e_2, e_3, e_6\}, \{e_1, e_2, e_4, e_5, e_6\}, \{e_3, e_6, e_7\}\big\}$ indicated by thick, dashed and zigzagged conflict graph edges before and after the construction step in the proof of Theorem 6.3.2, $j = 3$, $j' = 2$, $j'' = 1$. The hyperedge e_6 is the only one that belongs to three cliques, the hyperedges e'_6 and e''_6 belong to two and one clique, respectively. The number of cliques they belong to stays the same for all the other hyperedges.

Case 2: $p - J \leq i \leq p$**.** In this case a_k for $k \in K_i$ can have at most value $\frac{i-R}{p-R}$. Thus,

$$\begin{aligned} a'_i &= \sum_{k \in K_i} a_k \\ &\leq \sum_{k \in K_i} \frac{k-R}{p-R} \\ &= \frac{1}{p-R}\left(\sum_{k \in K_i} k - |K_i| \cdot R\right) \\ &= \frac{1}{p-R}(i - |K_i| \cdot R) \\ &\leq \frac{i-R}{p-R} \\ &= a_i, \end{aligned}$$

and, again, a_i cannot be increased.

Case 3: $p \leq i \leq \mathcal{Q}'$**.** Let $n = \left\lfloor \frac{i}{p} \right\rfloor$ and $j = n \mod p$, so $i = p \cdot n + j, n \in \mathbb{N}, 0 \leq j < p$. Then for $K_i = \{\underbrace{p, p, \ldots, p}_{n \text{ times}}, j\}$, $a'_i = n \cdot a_p + a_j = n + a_j$. We will

prove that this is the greatest possible value for a'_i. Note that as the a_i are monotonically increasing with i and equal for all $i \geq p$, the greatest possible value of a'_i can be achieved with $k \leq i$ for all $k \in K_i$. Thus, we can assume that $k \leq i$ for all $k \in K_i$. Further, observe that also for $i = p$, a_i can be written as $\frac{i-R}{p-R}$. Using this, we get

$$\begin{aligned} a'_i &= \sum_{k \in K_i} a_k \\ &\leq \sum_{k \in K_i} \max\left\{0, \frac{k-R}{p-R}\right\} \\ &\leq \frac{i - |K_i|R}{p-R} \\ &= n + \frac{j - (|K_i| - n)R}{p-R}. \end{aligned}$$

For $j = 0$, $|K_i|$ might be exactly n and the last term is equal to n. Otherwise $|K_i| \geq 1$ and the last term is bounded by $n + \frac{j-R}{p-R}$. In both cases this is $\leq n + a_j$.

Altogether, the three cases can be summarized to the following best possible strengthening of general clique family inequalities by Theorem 6.3.2. We fix $J = p - R$ as a smaller J cannot lead to a stronger inequality.

Theorem 6.3.3. *Let $\mathcal{Q}' \subseteq \mathcal{Q}$ be a set of at least three cliques for the hypergraph $G = (V, E)$, $2 \leq p \leq |\mathcal{Q}'|$, $p \in \mathbb{Z}$, $R := |\mathcal{Q}'| \mod p$. Then, for $i_e := |\{Q \in \mathcal{Q}' : e \in Q\}|$ and every $x \in P(SSP)$,*

$$\sum_{e \in E} \left(\left\lfloor \frac{i_e}{p} \right\rfloor + \max\left\{0, \frac{(i_e \mod p) - R}{p-R}\right\} \right) x_e \leq \left\lfloor \frac{|\mathcal{Q}'|}{p} \right\rfloor \qquad \text{(GCFOSI\_SSP)}$$

holds.

Note that for $p = 2$ and odd $|\mathcal{Q}'|$ this is exactly the inequality from Step 3 in Subsection 6.3.1.

6.3.4 Checking the Inequality Type

Given facets of a polytope, it is a non-trivial task to check whether they have a given type. One possibility is to enumerate all inequalities of some type that differ not just by a symmetry and then apply the facet classification method HUHFA described in Chapter 5 to check whether the facets belong to the same class as

Table 6.2: Number of possibilities for choices of a subset of maximal cliques $\mathcal{Q}'$, and coefficients in Inequality (GCFOSI_SSP) for $p = 2$, $R = 1$ in $G_{2,3}$. Choices are regarded as different if there is no symmetry that maps one of them to the other.

$\lvert\mathcal{Q}'\rvert$	#possibilities for $\mathcal{Q}'$	#possibilities for coefficients ($p = 2$, $R = 1$)
1	2	1
2	16	7
3	89	17
4	553	98
5	2994	429
6	15343	2094
7	70235	9001

one of the inequalities. However, the enumeration can lead to a very high number of inequalities and therefore be computationally not manageable. To give an impression of how many different inequalities can be generated, we show in Table 6.2 the number of possible choices of $\mathcal{Q}'$ and the number of possible coefficients in Inequality (GCFOSI_SSP), which is stated in Theorem 6.3.3, for $p = 2$, $R = 1$ for the hypergraph $G_{2,3}$. Two sets $\mathcal{Q}'_1, \mathcal{Q}'_2$ of maximal cliques are regarded as different possibilities for $\mathcal{Q}'$ and therefore both counted if there is no symmetry s from those 4608 defined in Section 5.5 such that for the vectors $x_1, x_2 \in \mathbb{R}^E$ defined by

$$(x_1)_e = |\{Q \in \mathcal{Q}'_1 : \ e \in Q\}|$$

and

$$(x_2)_e = |\{Q \in \mathcal{Q}'_2 : \ e \in Q\}|$$

$s(x_1) = x_2$ holds. Coefficient vectors $a_1, a_2 \in \mathbb{R}^E$ for a fixed $|\mathcal{Q}'|$ and therefore fixed right-hand side of Inequality (GCFOSI_SSP) are regarded as different if, again, there is no symmetry s from those defined in Section 5.5 such that $s(a_1) = a_2$ holds.

The numbers were found by enumeration. Since the numbers seem to grow exponentially with $|\mathcal{Q}'|$, we were only able to do this for $|\mathcal{Q}'| \leq 7$ in a reasonable computation time.

In the remainder of this subsection, we propose integer programming models that can be used to perform this check of inequality type in a different way for the most general inequality type we have derived, (GCFOSI_SSP). We will start with such a model which can be used to check whether an inequality for

some set packing problem has this type, and afterwards modify it so that it can also be used for hypergraph assignment problems, where, in contrast to set packing problems, equations are part of the IP formulation.

Let $G = (V, E)$ be a hypergraph. Let $\sum_{e \in E} a_e x_e \leq b$ with scalars $a_e, b \in \mathbb{R}$ and hyperedge variables x_e be an inequality. We want to check whether there exist a subset of at least three cliques $\mathcal{Q}' \subseteq \mathcal{Q}$, some $p \in \mathbb{Z}$ with $2 \leq p \leq |\mathcal{Q}'|$ and some multiplier of the given inequality f such that $f \cdot b \geq \left\lfloor \frac{|\mathcal{Q}'|}{p} \right\rfloor$, and for $R := |\mathcal{Q}'| \mod p$, $f \cdot a_e \leq \left\lfloor \frac{i_e}{p} \right\rfloor + \max\left\{0, \frac{(i_e \mod p) - R}{p - R}\right\}$ for $i_e := |\{Q \in \mathcal{Q}' : e \in Q\}|$. If this is the case, then $\sum_{e \in E} a_e x_e \leq b$ is implied by some inequality of the type (GCFOSI_SSP).

We will now present a mixed integer program that has a feasible solution if and only if such choices of $\mathcal{Q}' \subseteq \mathcal{Q}$ exist for fixed integers p and R. Then, the check can be done by solving the MIP for all pairs (p, R) where $2 \leq p \leq |\mathcal{Q}|$ and $0 \leq R < p$. Although this is not a very "elegant" way, for hypergraphs with a relatively small number of hyperedges, this works well. Usually, we get facets, which we want to analyze, only for small instances anyway. This formulation (with the enhancement for set partitioning type problems shown below) was also used to analyze the facets of the example of the HAP polytope in Section 5.5.

minimize	$\sum_{Q \in \mathcal{Q}:\ e \in Q} y_Q$		(SSPinequcheck)
subject to	$i_e = \sum_{Q \in \mathcal{Q}:\ e \in Q} y_Q$	$\forall e \in E$	(i)
	$i_e = p n_e + R_e$	$\forall e \in E$	(ii)
	$R_e \leq p - 1$	$\forall e \in E$	(iii)
	$\sum_{Q \in \mathcal{Q}} y_Q = pn + R$		(iv)
	$s_e \leq 1$	$\forall e \in E$	(v)
	$(p-1)s_e - p + 1 \leq R_e - R$	$\forall e \in E$	(vi)
	$R_e - R \leq (p-1)s_e$	$\forall e \in E$	(vii)
	$-Rs_e \leq (p-R)m_e$	$\forall e \in E$	(viii)
	$(p-R)m_e \leq (p-1-R)s_e$	$\forall e \in E$	(ix)
	$R_e - p + 1 + (p-R-1)s_e \leq (p-R)m_e$	$\forall e \in E$	(x)
	$(p-R)m_e \leq R_e - Rs_e$	$\forall e \in E$	(xi)
	$f\, b \geq n$		(xii)

$$f a_e \leq n_e + m_e \qquad \forall e \in E \quad \text{(xiii)}$$
$$y_Q, i_e, n_e, R_e, n, s_e, f \geq 0 \qquad \forall e \in E, Q \in \mathcal{Q} \quad \text{(xiv)}$$
$$y_Q \leq 1 \qquad \forall Q \in \mathcal{Q} \quad \text{(xv)}$$
$$y_Q, i_e, n_e, R_e, n, s_e \in \mathbb{Z} \qquad \forall e \in E, Q \in \mathcal{Q} \quad \text{(xvi)}$$

We introduce variables y_Q for all cliques $Q \in \mathcal{Q}$. y_Q is supposed to have value 1 if $Q \in \mathcal{Q}'$, and 0 otherwise. Further, let the variable i_e for every hyperedge $e \in E$ represent the number of selected cliques that contain e, i. e., $i_e = \sum_{Q \in \mathcal{Q}' :\, e \in Q} y_e$. The variable n_e will always have the value $\left\lfloor \frac{i}{p} \right\rfloor$ for all $e \in E$, and the variable R_e will be $i_e \mod p$ for all $e \in E$, such that $i_e = pn_e + R_e$. The variable n will have the value $\left\lfloor \frac{|\mathcal{Q}'|}{p} \right\rfloor$, i. e., $|\mathcal{Q}'| = np + R$. Further, we need to model the term $\max\left\{0, \frac{i_e \mod p - R}{p - R}\right\}$ in the coefficient of x_e, which will be done using the variable m_e for all $e \in E$.

The constraints work as follows.

- Equation (SSPinequcheck) (i) calculates the value of i_e based on the values of the variables y_Q.
- Constraints (SSPinequcheck) (ii) and (iii) combined with the non-negativity of R_e ensure the correct values of n_e and R_e.
- (SSPinequcheck) (iv) implies the correct choice of the value for n.
- s_e is an auxiliary binary variable for every $e \in E$ that has value 1 only if $R_e - R > 0$, i. e., $\max\{0, \frac{R_e - R}{p - R}\} = \frac{R_e - R}{p - R}$, and value 0 only if $\max\{0, \frac{R_e - R}{p - R}\} = 0$. This is modeled by Constraints (SSPinequcheck) (v)–(vii) and its non-negativity and integrality: For $s_e = 1$, Constraints (SSPinequcheck) (vi)–(vii) imply that $0 \leq R_e - R$, which is equivalent to $\max\{0, \frac{R_e - R}{p - R}\} = \frac{R_e - R}{p - R}$, and $R_e - R \leq p - 1$, which is always true since $R \geq 0$ and $R_e \leq p - 1$. For $s_e = 0$, Constraints (SSPinequcheck) (vi)–(vii) imply $-p + 1 \leq R_e - R$, which is always true since $R_e \geq 0$ and $R \leq p - 1$, and $R_e - R \leq 0$, which is equivalent to $\max\{0, \frac{R_e - R}{p - R}\} = 0$.
- Constraints (SSPinequcheck) (viii)–(xi) ensure the correct value of m_e for all $e \in E$ using the variable s_e: If $s_e = 1$, then Constraints (SSPinequcheck) (x)–(xi) imply that $m_e = \frac{R_e - R}{p - R}$, which was meant to be the value of m_e for $s_e = 1$, and Constraints (SSPinequcheck) (viii)–(ix) imply that $\frac{-R}{p - R} \leq m_e \leq \frac{p - 1 - R}{p - R}$, which are valid bounds on the value of m_e since $0 \leq R_e \leq p - 1$. If $s_e = 0$, then Constraints (SSPinequcheck) (viii)–(ix) imply that $m_e = 0$, which was meant to be the value of m_e for

$s_e = 0$, and Constraints (SSPinequcheck) (x)–(xi) imply that $R_e - p + 1 \leq (p-R)m_e \leq R_e$, which holds for $m_e = 0$ anyway since $0 \leq R_e \leq p-1$.

- Constraints (SSPinequcheck) (xii) and (xiii) ensure that Inequality (GCFOSI_SSP) given by the choice of variable values in this MIP indeed implies the inequality $\sum_{e \in E} a_e x_e \leq b$.
- Constraints (SSPinequcheck) (xiv) and (SSPinequcheck) (xvi) ensure the non-negativity and integrality of all variables. Inequality (SSPinequcheck) (xv) ensures that each clique is only once in $\mathcal{Q}'$ so that $\mathcal{Q}'$ is, indeed, a set.
- The objective function minimizes the number of chosen cliques. However, any other objective function can be chosen instead.

For set partitioning problems, where Inequalities (GCFOSI_SSP) also hold, the checking problem is a bit more complicated: Equations are present in the IP formulation, and some given inequality could be implied by (GCFOSI_SSP) only after the addition of multiples of some equations $\sum_{e \in \delta v} x_e = 1, v \in V$. To take this into account, we will use additional variables f_v for all $v \in V$ which will serve as multipliers of the corresponding equations, and consider how they change b and the a_e in the given inequality. The resulting MIP is the following.

minimize	$\sum_{Q \in \mathcal{Q}:\ e \in Q} y_Q$		(SPPinequcheck)
subject to	$i_e = \sum_{Q \in \mathcal{Q}:\ e \in Q} y_Q$	$\forall e \in E$	(i)
	$i_e = pn_e + R_e$	$\forall e \in E$	(ii)
	$R_e \leq p-1$	$\forall e \in E$	(iii)
	$\sum_{Q \in \mathcal{Q}} y_Q = pn + R$		(iv)
	$s_e \leq 1$	$\forall e \in E$	(v)
	$(p-1)s_e - p + 1 \leq R_e - R$	$\forall e \in E$	(vi)
	$R_e - R \leq (p-1)s_e$	$\forall e \in E$	(vii)
	$-Rs_e \leq (p-R)m_e$	$\forall e \in E$	(viii)
	$(p-R)m_e \leq (p-1-R)s_e$	$\forall e \in E$	(ix)
	$R_e - p + 1 + (p-R-1)s_e \leq (p-R)m_e$	$\forall e \in E$	(x)
	$(p-R)m_e \leq R_e - Rs_e$	$\forall e \in E$	(xi)

$$f\,b + \sum_{v \in V} f_v \geq n \tag{xii}$$

$$f\,a_e + \sum_{v \in V:\ e \in \delta(v)} f_v \leq n_e + m_e \qquad \forall e \in E \tag{xiii}$$

$$y_Q, i_e, n_e, R_e, n, s_e, f \geq 0 \qquad \forall e \in E, Q \in \mathcal{Q} \tag{xiv}$$

$$y_Q \leq 1 \qquad \forall Q \in \mathcal{Q} \tag{xv}$$

$$y_Q, i_e, n_e, R_e, n, s_e \in \mathbb{Z} \qquad \forall e \in E, Q \in \mathcal{Q} \tag{xvi}$$

The multipliers f_v appear in Constraints (xii) and (xiii).

6.3.5 Separation IP

The idea of the MIPs in Section 6.3.4 can be also used to separate Inequalities (GCFOSI_SSP), again for fixed p and R. Constraints (SSPinequcheck) (i)–(xi) basically describe Inequalities (GCFOSI_SSP), they can be also used for the separation. To check whether some $x \in \mathbb{R}^E$ can be separated by such an inequality, we do not need Inequalities (SSPinequcheck) (xii)–(xiii). Instead, we need to know whether x violates the corresponding Inequality (GCFOSI_SSP). In the following IP we do this using the objective function value. Its optimal value is greater than 0 if and only if x violates some Inequality (GCFOSI_SSP) with given p and R with $2 \leq p \leq |\mathcal{Q}|$ and $0 \leq R < p$.

maximize $\sum_{e \in E} x_e(n_e + m_e) - n$ (GCFOSI_SSPseparation)

subject to

$$i_e = \sum_{Q \in \mathcal{Q}:\ e \in Q} y_Q \qquad \forall e \in E \tag{i}$$

$$i_e = p n_e + R_e \qquad \forall e \in E \tag{ii}$$

$$R_e \leq p - 1 \qquad \forall e \in E \tag{iii}$$

$$\sum_{Q \in \mathcal{Q}} y_Q = pn + R \tag{iv}$$

$$s_e \leq 1 \qquad \forall e \in E \tag{v}$$

$$(p-1)s_e - p + 1 \leq R_e - R \qquad \forall e \in E \tag{vi}$$

$$R_e - R \leq (p-1)s_e \qquad \forall e \in E \tag{vii}$$

$$-Rs_e \leq (p-R)m_e \qquad \forall e \in E \tag{viii}$$

$$(p-R)m_e \leq (p-1-R)s_e \qquad \forall e \in E \tag{ix}$$

$$R_e - p + 1 + (p-R-1)s_e \leq (p-R)m_e \qquad \forall e \in E \tag{x}$$

$$(p-R)m_e \leq R_e - Rs_e \qquad \forall e \in E \qquad \text{(xi)}$$

$$y_Q, i_e, n_e, R_e, n, s_e \geq 0 \qquad \forall e \in E, Q \in \mathcal{Q} \qquad \text{(xii)}$$

$$y_Q \leq 1 \qquad \forall Q \in \mathcal{Q} \qquad \text{(xiii)}$$

$$y_Q, i_e, n_e, R_e, n, s_e \in \mathbb{Z} \qquad \forall e \in E, Q \in \mathcal{Q} \qquad \text{(xiv)}$$

Inequalities (OSI_SSP) developed in Step 3 in Subsection 6.3.1 can be also separated as a subset of Chvátal-Gomory cuts with coefficients 0 and $\frac{1}{p}$ as we have shown there. Especially for the case of coefficients from $\left\{0, \frac{1}{2}\right\}$, these cuts can be separated effectively in practice although this is theoretically $\mathcal{NP}$-complete [Koster et al., 2009]. All the facets of P(HAP) for the hypergraph $G_{2,3}$ that we could find to be of the type (GCFOSI_SSP) using the IP (SPPinequcheck) are of the inequality type (OSI_SSP) with $p = 2$ and $R = 1$.

Chapter 7

Local Search with Network CoCo

In this chapter, we develop an exact combinatorial solution method for the HAP.

We start in Section 7.1 with a very large-scale neighborhood search. The idea will be to subdivide the hyperassignments into groups such that the best hyperassignment in a group can be found in polynomial time while each group contains an exponential number of hyperassignments in a complete partitioned hypergraph w. r. t. the number of parts.

Each group will have a polynomial number of neighbor groups. We will move from one group to a neighbor group that contains a hyperassignment with minimum cost from all those groups. This polynomial time step will move to a group containing the minimum cost hyperassignment from an exponentially sized set of hyperassignments.

If the local search reaches a local minimum, which means that it finds a solution such that none of its neighbors has a lower cost, it has to be checked whether the current solution is also a global minimum and thus optimal, and if this is not the case, find how to continue the search. To this end, we will employ the composite columns method ("CoCo") published in [Balas and Padberg, 1975]. It allows to start at some integral solution of a set partitioning problem and then to perform a combination of steps of the simplex algorithm that find a solution with lower cost if it exists. Otherwise the method proves that the current solution is optimal.

To perform such a step of the composite columns method, not just the current solution but a basis for the simplex method is needed. Then, certain vectors, directions of basis changes as well as reduced costs, have to be calculated. This is typically done by algebraic computations that might be computationally unstable and do not exploit the combinatorial structure of the problem. For flow problems in graphs, however, the network simplex algorithm [Orlin,

1997] allows to carry out these computations combinatorially. We will apply this method also for the HAP.

To assemble this local minimum escape, we proceed as follows. In Section 7.2 we show an analysis of the basis matrices for the LP relaxation of (HAP). For graphs, it is known that bases correspond to spanning trees, which is the foundation of the network simplex method. This is not so easy for the hypergraph case. We will, however, show that in certain cases the bases are such that the results for graphs can be still adapted. This is the subject of Section 7.3. There, we show that sometimes the basis can be transformed such that the result is, indeed, a spanning tree. For such bases we will show in Section 7.4 how the calculations for a step of the simplex algorithm can be performed using the combinatorial methods which are known for graphs. For every hyperassignment, i. e., vertex of P(HAP), such a basis can be easily constructed, as is shown in Section 7.5.

In Section 7.6 we sum up the method.

7.1 Very Large-Scale Neighborhood

We will partition the set of hyperassignments such that a hyperassignment with minimum cost from the set of all the hyperassignments in one partition can be found combinatorially in polynomial time. The partitions will be called hyperassignments respecting a certain vertex grouping since they can be described by partitions of the vertices in the parts of a partitioned hypergraph. The number of hyperassignments respecting each vertex grouping will be exponential in the number of parts of the hypergraph. We will then order the vertex groupings such that they can be described and visited in a systematic way. Since the number of different vertex groupings is unfortunately exponential in the number of parts of size ≥ 2 of the hypergraph, too, we will have to stick to a heuristic search of a vertex grouping containing an optimum solution to the hypergraph assignment problem. This heuristic will be a local search.

Definition 7.1.1. Let $G = (U, V, E)$ be a partitioned hypergraph with parts $\{U_1, \dots, U_p\}$ and $\{V_1, \dots, V_q\}$ on the U-side and the V-side, respectively. A partition $\mathscr{G} \subseteq 2^{U \cup V}$ of $U \cup V$, i. e., a set of pairwise disjoint subsets of $U \cup V$ with union $U \cup V$, is called a *vertex grouping* of G if the following two requirements hold:

1. For every vertex subset $W \in \mathscr{G}$ there exists a part Π such that $W \subseteq \Pi$, i. e., for every part Π there exists a subset of $\mathscr{G}$ which is a partition of Π.

2. For every $n \in \mathbb{N}$,

$$|\{W \in \mathcal{G} : W \subseteq U, |W| = n\}| = |\{W \in \mathcal{G} : W \subseteq V, |W| = n\}|,$$

i. e., there exists the same number of n-element subsets of U and of V in $\mathcal{G}$.

A hyperedge $e \in E$ *respects a vertex grouping* $\mathcal{G}$ if and only if $e \cap U \in \mathcal{G}$ and $e \cap V \in \mathcal{G}$. A hyperassignment H in G respects a vertex grouping $\mathcal{G}$ if and only if every hyperedge $e \in H$ respects $\mathcal{G}$. Further, every hyperassignment H in G gives rise to a unique vertex grouping $\mathcal{G}(H) := \{e \cap U, e \cap V : e \in H\}$ that it respects.

For an example of a vertex grouping, see Figure 7.1. The unique vertex grouping $\mathcal{G}(H)$ that a hyperassignment H respects gives rise to an extended formulation for the HAP with extra variables $z_{\Pi,S}$ for all parts Π and all the possible partitions S of Π.

$$\begin{aligned}
\underset{x \in \mathbb{R}^E}{\text{minimize}} \quad & \sum_{e \in E} c_E(e) x_e && && \text{(HAP_vgroups)} \\
\text{subject to} \quad & \sum_{\text{partitions } S \text{ of } \Pi} z_{\Pi,S} = 1 && && \forall \text{parts } \Pi \\
& && && \text{(i)} \\
& \sum_{e \in E:\ e \cap U = W \text{ or } e \cap V = W} x_e = \sum_{\text{partitions } S \text{ of } \Pi : W \in S} z_{\Pi,S} && && \forall W \subseteq \Pi, \forall \text{parts } \Pi \\
& && && \text{(ii)} \\
& x, z \geq 0 && && \text{(iii)} \\
& x, z \in \mathbb{Z}^E. && && \text{(iv)}
\end{aligned}$$

Equations (HAP_vgroups) (i) describe the selection of exactly one partition S for each part Π. Equations (HAP_vgroups) (ii) ensure that the hyperedges in the hyperassignment respect a vertex grouping that contains the sets from S.

This extended formulation (HAP_vgroups) is correct, but the projection of the polytope of the LP relaxation onto the hyperedge variables is exactly $P_{\text{LP}}(\text{HAP})$, so that (HAP_vgroups) does not strengthen (HAP). This can be shown as follows. Let $v \in U \cup V$ be some vertex of the partitioned hypergraph $G = (U, V, E)$, and let Π be the part that contains v. Summing up Equations (HAP_vgroups) (ii) for all W that contain v leads to the sum of all hyperedge variables $e \in \delta(v)$ on the left-hand side, and the sum of all variables $z_{\Pi,S}$ for partitions S of Π on the right-hand side. Substitution of Equation

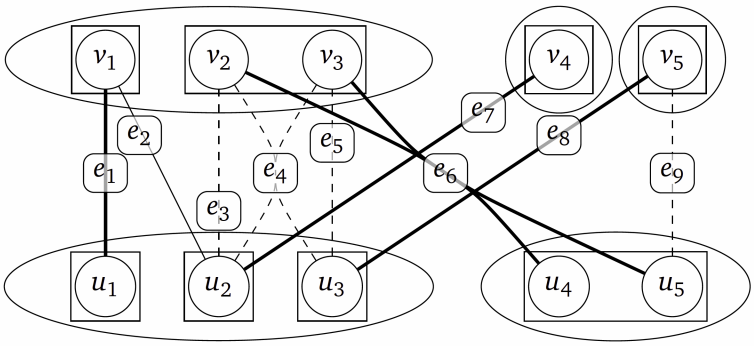


Figure 7.1: Partitioned hypergraph G with vertex grouping $\mathcal{G} = \big\{\{u_1\},\{u_2\},\{u_3\},\{u_4,u_5\},\{v_1\},\{v_2,v_3\},\{v_4\},\{v_5\}\big\}$ indicated by rectangles. The hyperassignment $\{e_1,e_6,e_7,e_8\}$ (drawn with thick lines), the contained hyperedges, and also hyperedge e_2 (thin solid line) respect $\mathcal{G}$. The other hyperedges e_3,e_4,e_5,e_9 (drawn with dashed lines) do not respect the vertex grouping $\mathcal{G}$.

(HAP_vgroups) (i) on the right-hand side gives exactly Equation (HAP) (i). On the other hand, for a feasible solution of the LP relaxation of (HAP), choosing the values of z according to (HAP_vgroups) (ii) gives a feasible solution for the LP relaxation of (HAP_vgroups).

We now show how a vertex grouping restricts the feasible solutions of the HAP to a set of hyperassignments which is easier to analyze.

Lemma 7.1.2. *Let $\mathcal{G}$ be a vertex grouping of the partitioned hypergraph $G = (U,V,E)$, and let $c_E : E \to \mathbb{R}$ be a cost function. A minimum cost hyperassignment in G w. r. t. c_E which respects $\mathcal{G}$, i. e., a hyperassignment H^* in G such that*

$$c_E(H^*) = \min\{c_E(H) : \text{ } H \text{ is a hyperassignment in } G \text{ which respects } \mathcal{G}\},$$

or the information that no such hyperassignment exists can be obtained in polynomial time.

Proof. We will prove that the problem of finding a minimum cost hyperassignment that respects a fixed vertex grouping is equivalent to an assignment problem in a bipartite graph with $|\mathcal{G}|$ vertices. To this end, we will construct a bipartite graph $G' = (U',V',E')$ with this number of vertices and a cost function $c_{E'} : E' \to \mathbb{R}$, and show that there is a cost-preserving bijection b between hyperassignments H in G that respect $\mathcal{G}$ and assignments H' in G'.

Define

$$\begin{aligned}
U' &:= \{u'(W) :\ W \in \mathscr{G},\ W \subseteq U\},\\
V' &:= \{v'(W) :\ W \in \mathscr{G},\ W \subseteq V\},\\
e' &:= \{u'(e \cap U), v'(e \cap V)\} && \forall e \in E \text{ that respect } \mathscr{G},\\
E' &:= \left\{e' :\ e \in E,\ e \text{ respects } \mathscr{G}\right\},\\
c_{E'}(e') &:= c_E(e) && \forall e \in E \text{ that respect } \mathscr{G}.
\end{aligned}$$

Note that by definition of a hyperedge that respects a vertex grouping, $e \cap U, e \cap V \in \mathscr{G}$, so all $e' \in E$ are pairs of a vertex in U' and a vertex in V' and E' is thus well-defined.

Now, define

$$\begin{aligned}
b : \{\text{hyperassignments } H \text{ in } G :\ H \text{ respects } \mathscr{G}\} &\to \left\{\text{assignments } H' \text{ in } G'\right\},\\
H &\mapsto H' := \{e' :\ e \in H\}.
\end{aligned}$$

Obviously, b is a bijection, and $c_E(H) = c_{E'}(b(H))$, i. e., it is cost-preserving. For an example of the construction, see Figure 7.2.

It remains to show that it maps hyperassignments in G that respect $\mathscr{G}$ to assignments in G' and vice versa.

Let H be a $\mathscr{G}$ respecting hyperassignment in G. We have to proof that for every $u'(W) \in U'$ there exists a unique edge $e' \in H' = b(H)$ such that $e' \cap U' = \{u'(W)\}$ (analogously for $v'(W) \in V'$). By definition of a hyperassignment and since it respects $\mathscr{G}$, there exists a unique $e \in E$ such that $e \cap U = W$. By construction of b, e then is the unique edge in H that is mapped by b to $\{u'(W), v'\}$ for some $v' \in V'$.

On the other hand, if for some set $E \in H$, $H' = \{e' : e \in H\}$ is an assignment, then $b^{-1}(H') = H$ must be a hyperassignment that respects $\mathscr{G}$. The statement about $\mathscr{G}$ is clear. For a vertex $u \in U$ (analogously for $v \in V$) there exists by definition of a vertex grouping a unique $W \in \mathscr{G}$ such that $u \in W$. Since H' is an assignment, it contains exactly one $e' \in E'$ such that $u'(W) \in e'$. Therefore, e is the unique hyperedge in H that is incident to u. □

Observe that the number of hyperassignments that respect a given vertex grouping can have the size $O(|\mathscr{G}|!)$ if all possible hyperedges are in the hyperedge set of the hypergraph and therefore G' from the proof of Lemma 7.1.2 can be a complete bipartite graph. Thus, the polynomial time method from the Lemma 7.1.2 can be used to find the best of exponentially many hyperassignments—those that respect $\mathscr{G}$. This idea gives rise to a very large-scale neighborhood search (see [Ahuja et al., 2002] for a survey on this topic).

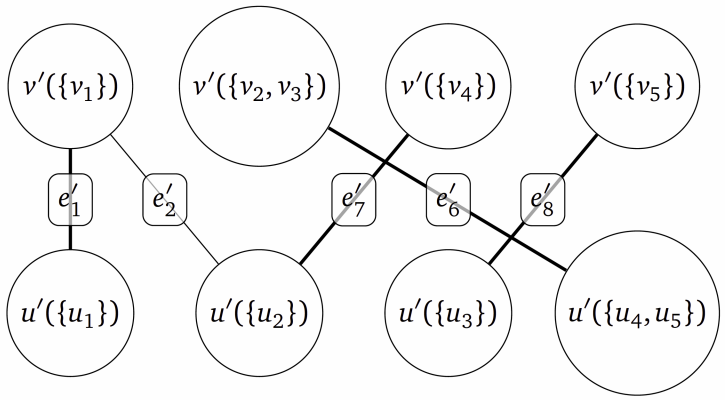


Figure 7.2: Bipartite graph G' as constructed in proof of Lemma 7.1.2 from the partitioned hypergraph G with the vertex grouping $\mathcal{G}$ in Figure 7.1. The edges belonging to the assignment $b(H) = \{e'_1, e'_6, e'_7, e'_8\}$ are drawn with thick lines.

The idea is to move from a vertex grouping to a next vertex grouping such that the minimum cost hyperassignment in the next grouping is better than the minimum cost of the previous vertex grouping. Such a step will screen exponentially many hyperassignments—all hyperassignments that respect the new vertex grouping—in polynomial time.

We will check several vertex grouping candidates before we move to the next one. Either, we subdivide two vertex sets $U' \subseteq U$ and $V' \subseteq V$, from the grouping, or we unite the two sets $U', U'' \subseteq U$ and $V', V'' \subseteq V$ from one part, respectively. The next definition explains this in more detail.

Definition 7.1.3. Let $G = (U, V, E)$ be a partitioned hypergraph and $\mathcal{G}$ and $\mathcal{G}'$ vertex groupings of G. We call $\mathcal{G}$ a *refinement* of $\mathcal{G}'$ if $|\mathcal{G}' \setminus \mathcal{G}| = 1$, $|\mathcal{G} \setminus \mathcal{G}'| = 2$ and the union of the two sets in $\mathcal{G} \setminus \mathcal{G}'$ is the set in $\mathcal{G}' \setminus \mathcal{G}$. The vertex grouping $\mathcal{G}'$ is then called a *coarsening* of the vertex grouping $\mathcal{G}$.

We call moving from one vertex grouping to one of its refinements a *refinement step*, and moving from one vertex grouping to one of its coarsenings a *coarsening step*. Figure 7.3 shows all the possible vertex groupings as well as coarsening and refinement steps for the hypergraph from Figure 7.1. For $G_{2,3}$, $G_{2,4}$, and $G_{2,5}$, which allow many more vertex groupings, Figures 7.4, 7.5, and 7.6 give an impression of which vertex groupings can be reached by a coarsening or refinement step from each other, respectively.

Note that there is no coarsening of the vertex grouping which is the set of all parts (if the part sizes are such that this is, indeed, a vertex grouping, i. e., the number of parts of each size is the same on the U- and the V-side), and that there

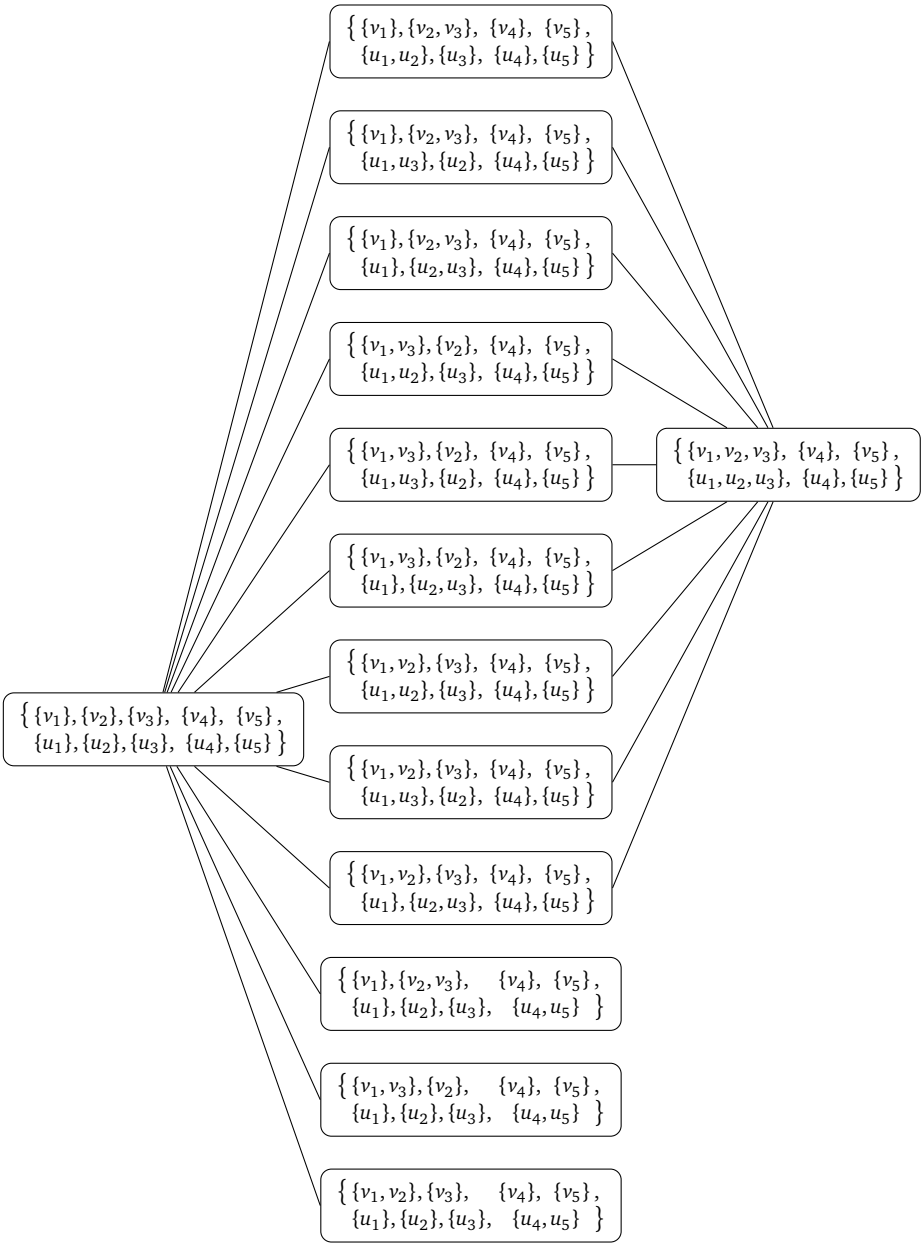


Figure 7.3: All possible vertex groupings for the hypergraph G from Figure 7.1. Connecting lines indicate all the possible coarsening (from the left to the right) and refinement (from the right to the left) steps.

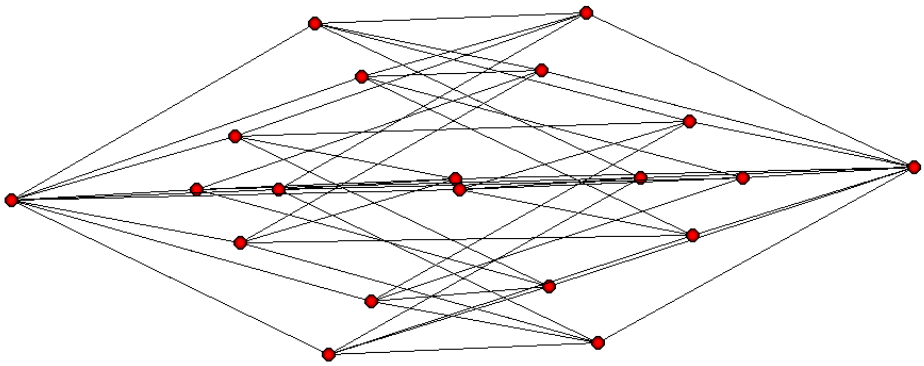


Figure 7.4: All possible vertex groupings for the hypergraph $G_{2,3}$ indicated by red circles, and the possible coarsening and refinement steps indicated by connections of the circles.

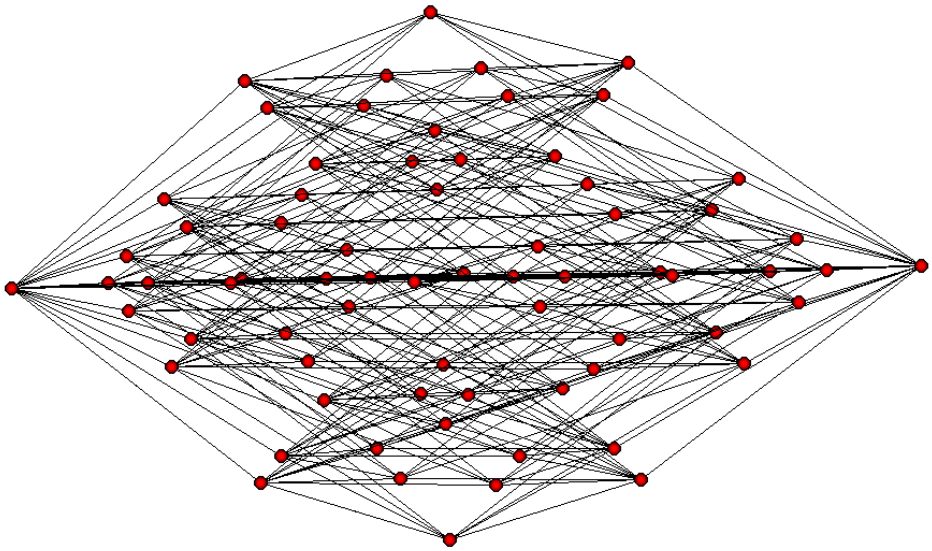


Figure 7.5: All possible vertex groupings for the hypergraph $G_{2,4}$ indicated by red circles, and the possible coarsening and refinement steps indicated by connections of the circles.

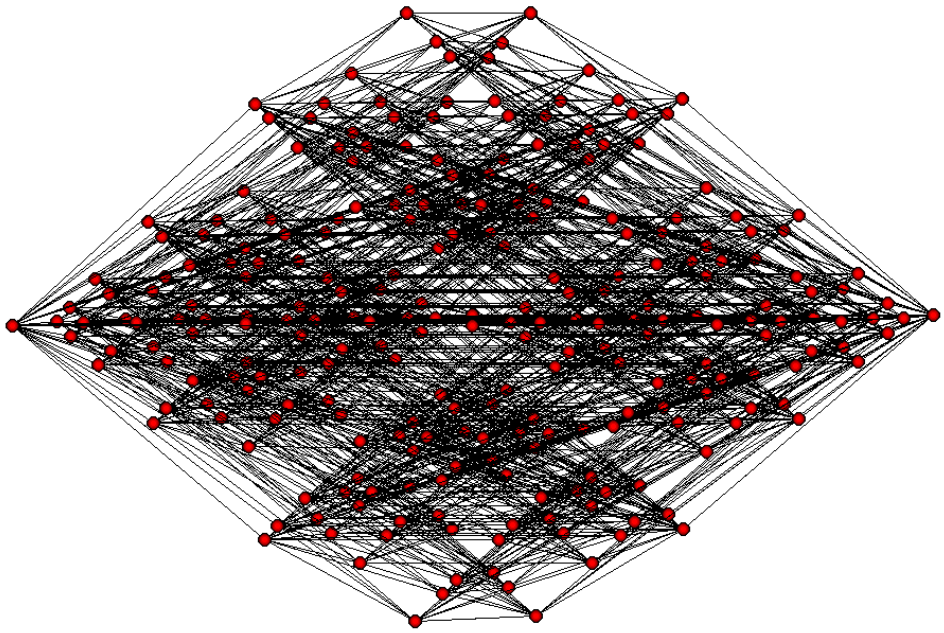


Figure 7.6: All possible vertex groupings for the hypergraph $G_{2,5}$ indicated by red circles, and the possible coarsening and refinement steps indicated by connections of the circles.

is no refinement of the vertex grouping consisting of only one-element sets. They are the "*coarsest*" and "*finest*" vertex groupings, respectively. Each vertex grouping can be reached from the finest vertex grouping by only coarsening steps. The number of steps is the difference between the number of sets in the two groupings divided by two. The maximum number of steps, i. e., the distance between the coarsest and the finest vertex grouping, in a hypergraphs with the same part sizes on both sides is the number of vertices on one side minus the number of parts on one side.

Algorithm 7.1.1: Given a hyperassignment H, checks whether a hyperassignment H' with a lower cost exists such that $\mathcal{G}(H')$ is a refinement/coarsening of $\mathcal{G}(H)$.

Data: Partitioned hypergraph $G = (U, V, E)$, cost function $c_E : E \to \mathbb{R}$, hyperassignment H in G, and Direction $\in \{0, 1\}$ indicating whether a refinement or coarsening has to be found.

Result: Hyperassignment H' in G with $c_E(H') < c_E(H)$ such that $\mathcal{G}(H')$ is a refinement (if Direction $= 0$) or coarsening (if Direction $= 1$) of $\mathcal{G}(H)$ if such an H' exists. Returns H otherwise.

```
1  BestNeighbor ← H
2  CurrentNeighbor ← H
3  if Direction = 0 then
4  |  VertexGroupings ← set of all refinements of 𝒢(H)
5  else
6  |  VertexGroupings ← set of all coarsenings of 𝒢(H)
7  foreach 𝒢 ∈ VertexGroupings do     // if G has p parts on the
   U-side and q parts on the V-side, |VertexGroupings| ≤ pq
8  |  CurrentNeighbor ← hyperassignment in G with minimum cost w. r. t.
   |  c_E that respects 𝒢 if it exists   // can be found in polynomial
   |  time using, e. g., the Hungarian method
9  |  if c_E(CurrentNeighbor) < c_E(BestNeighbor) then
10 |  |  BestNeighbor ← CurrentNeighbor
11 return BestNeighbor
```

Algorithm 7.1.1 performs a best possible refinement/coarsening step. Given a partitioned graph G, a cost function, and a hyperassignment H in G, it finds a hyperassignment H' with cost lower than H (if it exists) such that $\mathcal{G}(H')$ is a refinement/coarsening of $\mathcal{G}(H)$. If more than one such hyperassignment exists,

it chooses the one with the lowest cost.

A very large-scale neighborhood local search can now, for example, start at the finest vertex grouping and then in each step find the coarsening with the lowest cost of a hyperassignment respecting this coarsening. If this cost is lower than that of the current vertex grouping, we can move to this grouping and continue from there. Otherwise we can continue with refinement steps. The highest possible number of coarsening or refinement steps that can be done in a row is the distance between the coarsest and finest vertex grouping. A vertex grouping where neither a coarsening nor a refinement steps leads to a solution with lower cost is called a *local minimum*. To find a better solution than the local minimum, we have to restart the local search with some new starting point that hopefully will not lead to the same local minimum, or use a different method to find a better solution than the local minimum. We will propose such a method in the following sections.

To get an impression on the distribution of local minima, we enumerated the optimal values of hyperassignments for each vertex grouping in the complete partitioned hypergraph $G_{2,n}$ having n parts of size 2 on the U-side and on the V-side, and counted the number of local minima for different n. The mean number of local minima for different cost functions and different n for 100 random instances each are depicted in Figure 7.7. As cost function types, we have chosen all those that were already used for the calculations in Chapter 4, i. e., i. i. d. uniform random variables on $[0,1]$, i. i. d. exponential random variables with mean 1, and the regularity rewarding cost functions with different penalty values. We can see that the number of local minima seems to rise exponentially for most of the cost functions. How fast the number increases, depends on the cost function type.

A local and global minimum can be as far away as the distance between the coarsest and finest vertex grouping. This is shown in the following example.

Example 7.1.4. Let $G_{2,n} = (U, V, E)$, $n \geq 2$ be the complete partitioned hypergraph with parts $U_1 = \{u_1, u_1'\}$, $U_2 = \{u_2, u_2'\}$, …, $U_n = \{u_n, u_n'\}$ on the U-side and parts $V_1 = \{v_1, v_1'\}$, $V_2 = \{v_2, v_2'\}$, …, $V_n = \{v_n, v_n'\}$ on the V-side. Define the cost function c_E as follows:

$$c_E(e) = \begin{cases} 0 & \text{if } e \in \{U_1 \cup V_2, U_2 \cup V_3, \ldots, U_{n-1} \cup V_n, U_n \cup V_1\} \\ 1 & \text{if } e \in \left\{\{u_1, v_1\}, \{u_1', v_1'\}, \ldots, \{u_n, v_n\}, \{u_n', v_n'\}\right\} \\ 2n+1 & \text{otherwise.} \end{cases}$$

Then the minimum cost hyperassignment is obviously the set of all hyperedges with cost 0 and respects the coarsest vertex grouping. The only hyperassignment with cost $2n$ is the hyperassignment that consists of all the edges with

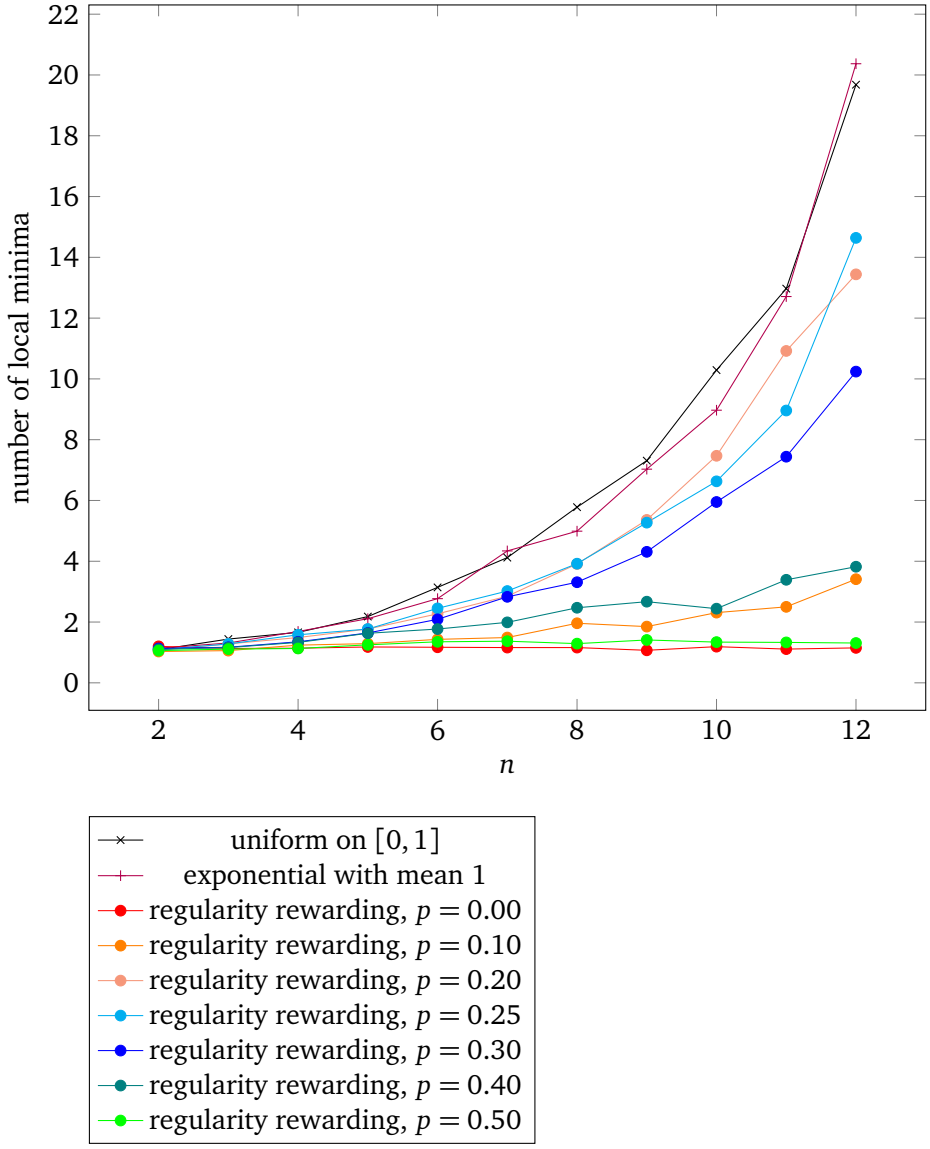


Figure 7.7: Mean number of local minima in $G_{2,n}$ for 100 calculations for each value of n and each type of random cost function.

cost 1. This hyperassignment respects the finest vertex grouping. Since all other hyperassignments have cost at least $2n+1$, it is a local minimum.

On the other hand, it is possible that there exists only one local minimum, which then also has to be the global minimum.

Example 7.1.5. Consider again $G_{2,n} = (U, V, E)$ with parts $U_1 = \{u_1, u_1'\}$, $U_2 = \{u_2, u_2'\}$, …, $U_n = \{u_n, u_n'\}$ on the U-side and parts $V_1 = \{v_1, v_1'\}$, $V_2 = \{v_2, v_2'\}$, …, $V_n = \{v_n, v_n'\}$ on the V-side. Let $I, J \subseteq N := \{1, \dots, n\}$ be some index sets with the same cardinality $|I| = |J| = n-k$ for some $k \in \{0, \dots, n\}$. Assume that there exists some $M > 0$ such that for the cost function $c_E : E \to \mathbb{R}$ the following four conditions hold.

- $0 \leq c_E(e) < \frac{M}{2n}$ if $e = \{u, v\}, u \in U_i, v \in V_j$ with $i \in N \setminus I$ and $j \in N \setminus J$ (edges of type 1).
- $0 \leq c_E(e) < \frac{M}{n}$ if $e = U_i \cup V_j$ with $i \in I$ and $j \in J$ (proper hyperedges of type 1).
- $M \leq c_E(e) < M + \frac{M}{2n}$ if $e = \{u, v\}, u \in U_i, v \in V_j$ with $i \in I$ or $j \in J$ (edges of type 2).
- $2M \leq c_E(e) < 2M + \frac{M}{n}$ if $e = U_i \cup V_j$ with $i \in N \setminus I$ or $j \in N \setminus J$ (proper hyperedges of type 2).

For a vertex grouping $\mathscr{G}$, let

$$k(\mathscr{G}) := \min\left\{|\{i \in N \setminus I : \{u_i\}, \{u_i'\} \in \mathscr{G}\}|, |\{j \in N \setminus J : \{v_j\}, \{v_j'\} \in \mathscr{G}\}|\right\},$$
$$l(\mathscr{G}) := \min\left\{|\{i \in I : U_i \in \mathscr{G}\}|, |\{j \in J : V_j \in \mathscr{G}\}|\right\}.$$

For a fixed vertex grouping, which implies a fixed number of edges and proper hyperedges, a minimum cost hyperassignment that respects the vertex grouping will use as many as possible edges of type 1 (each edge of type 2 is more expensive than each edge of type 1) and as many as possible proper hyperedges of type 1 (each proper hyperedge of type 2 is more expensive than each proper hyperedge of type 1). Then a minimum cost hyperassignment H that respects $\mathscr{G}$ will contain $2 \cdot k(\mathscr{G})$ edges of type 1 and $l(\mathscr{G})$ proper hyperedges of type 1. For its cost therefore

$$(n - k(\mathscr{G}) - l(\mathscr{G})) \cdot 2M \leq c_E(H) < (n - k(\mathscr{G}) - l(\mathscr{G}) + 1) \cdot 2M$$

holds. Thus, a minimum cost hyperassignment that respects a certain vertex grouping $\mathscr{G}$ has cost less than a minimum cost hyperassignment that respects another vertex grouping $\mathscr{G}'$ if and only if $k(\mathscr{G}') + l(\mathscr{G}') < k(\mathscr{G}) + l(\mathscr{G})$.

Obviously, for the vertex grouping

$$\mathcal{G}^* := \{U_i, V_j : \ i \in I, j \in J\} \cup \left\{\{u_i\}, \{u_i'\}, \{v_j\}, \{v_j\} : \ i \in N \setminus I, j \in N \setminus J\right\},$$

$k(\mathcal{G}^*) = k$ and $l(\mathcal{G}^*) = n - k$ so that $k(\mathcal{G}^*) + l(\mathcal{G}^*)$ is equal to n and therefore maximal. For every other vertex grouping $\mathcal{G}$, $k(\mathcal{G}) + l(\mathcal{G}) < n$. Therefore, the global minimum, i. e., the vertex grouping that contains a hyperassignment with minimum cost, is at the vertex grouping $\mathcal{G}^*$.

We will show now that all other vertex groupings $\mathcal{G} \neq \mathcal{G}^*$ are not local minima. This can be done by finding a coarsening or refinement $\mathcal{G}'$ with $k(\mathcal{G}')+ l(\mathcal{G}') > k(\mathcal{G}) + l(\mathcal{G})$. W. l. o. g., assume that

$$k(\mathcal{G}) = |\{i \in N \setminus I : \ \{u_i\}, \{u_i'\} \in \mathcal{G}\}| \leq |\{j \in N \setminus J : \ \{v_j\}, \{v_j'\} \in \mathcal{G}\}|,$$

and therefore

$$l(\mathcal{G}) = |\{i \in I : \ U_i \in \mathcal{G}\}| \leq |\{j \in J : \ V_j \in \mathcal{G}\}|.$$

We distinguish four cases. At least one of the cases has to hold if $k(\mathcal{G}) \neq k$ and $l(\mathcal{G}) \neq n - k$ (otherwise $\mathcal{G} = \mathcal{G}^*$).

Case 1: $k(\mathcal{G}) < k$ **and** $|\{j \in J : \ V_j \in \mathcal{G}\}| - |\{i \in I : \ U_i \in \mathcal{G}\}| \geq 1$. There exists some $i \in N \setminus I$ such that $U_i \in \mathcal{G}$. Choose some $j \in J$ be such that $V_j \in \mathcal{G}$. Then the vertex grouping $\mathcal{G}' := \left(\mathcal{G} \cup \left\{\{u_i\}, \{u_i'\}, \{v_j\}, \{v_j'\}\right\}\right) \setminus \{U_i, V_j\}$ is a refinement of $\mathcal{G}$ and $k(\mathcal{G}') = k(\mathcal{G}) + 1$, $l(\mathcal{G}') = l(\mathcal{G})$.

Case 2: $k(\mathcal{G}) < k$ **and** $|\{j \in J : \ V_j \in \mathcal{G}\}| = |\{i \in I : \ U_i \in \mathcal{G}\}|$. There exists some $i \in N \setminus I$ such that $U_i \in \mathcal{G}$ and some $j \in N \setminus J$ such that $U_j \in \mathcal{G}$. Then the vertex grouping $\mathcal{G}' := \left(\mathcal{G} \cup \left\{\{u_i\}, \{u_i'\}, \{v_j\}, \{v_j'\}\right\}\right) \setminus \{U_i, V_j\}$ is a refinement of $\mathcal{G}$ and $k(\mathcal{G}') = k(\mathcal{G}) + 1$, $l(\mathcal{G}') = l(\mathcal{G})$.

Case 3: $l(\mathcal{G}) < n - k$ **and**

$$|\{j \in N \setminus J : \ \{v_j\}, \{v_j'\} \in \mathcal{G}\}| - |\{i \in N \setminus I : \ \{u_i\}, \{u_i'\} \in \mathcal{G}\}| \geq 1.$$

We can find a coarsening analogously to Case 1.

Case 4: $l(\mathcal{G}) < n - k$ **and**

$$|\{j \in N \setminus J : \ \{v_j\}, \{v_j'\} \in \mathcal{G}\}| = |\{i \in N \setminus I : \ \{u_i\}, \{u_i'\} \in \mathcal{G}\}|.$$

We can find a coarsening analogously to Case 2.

7.2 Basis Matrices in the Simplex Method

To find a way to escape from local minima, we will employ a combination of steps of the simplex method, which can be done combinatorially. To this end, we begin in this section with observations on the basis matrices, which are important in the simplex method.

Let $A \in \mathbb{R}^{(U \dot\cup V) \times E}$ be the coefficient matrix of the standard IP formulation (HAP) of the HAP for a bipartite hypergraph $G = (U, V, E)$. The entries of $A = (a_{ve})_{v \in U \cup V, e \in E}$ are

$$a_{ve} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise.} \end{cases}$$

Thus, A is the incidence matrix of G.

Note that if a hyperedge e is the disjoint union of other hyperedges, the sum of the corresponding columns of A is equal to $A_{\cdot e}$ and vice versa, i. e.,

$$e = e_1 \dot\cup e_2 \dot\cup \ldots \dot\cup e_k \Leftrightarrow A_{\cdot e} = A_{\cdot e_1} + A_{\cdot e_2} + \ldots + A_{\cdot e_k}.$$

Since all hyperedges in a bipartite hypergraph cover the same number of vertices in U and V, a set of pairwise disjoint hyperedges that covers all but one vertex from U and V exactly once covers also the remaining vertex exactly once and is thus necessarily a hyperassignment. Therefore, one of Equations (HAP) (i) is redundant and can be removed. For the matrix A this means that it does not have full row rank, its rows are linearly dependent. This is true since the following equation holds.

$$\sum_{u \in U} A_{u\cdot} - \sum_{v \in V} A_{v\cdot} = 0$$

The entry for $e \in E$ of the row vector that results on the left-hand side of the equation is $\sum_{u \in e \cap U} 1 - \sum_{v \in e \cap V} 1$, which is equal to 0 since every hyperedge by definition contains the same number of vertices from the two sets U and V. On the other hand, at least for the case where all possible edges are in E, we know from Theorem 6.1.1 that no other equalities can be redundant. We will stick to this case in this section since having all possible edges in E will simplify the notation in the further analysis and adding edges that are not in G with very high cost does not change the HAP.

Then, if we apply the simplex method to the LP relaxation of (HAP), a basis consists of one less than the number of Equations (HAP) (i), i. e., $|U| + |V| - 1$ columns of A. Columns that form a basis have to be linearly independent. We will explore which $(|U| + |V| - 1)$-element subsets B of the hyperedge set E are

such that the corresponding columns $A_{\cdot e}$, $e \in B$ of A are linearly independent, and how steps of the simplex method for certain bases can be understood combinatorially. For short, we will also write that certain hyperedges are linearly dependent or independent or form a basis if we mean that the corresponding columns are linearly dependent or independent or form a basis, respectively.

In this section, we will state a few necessary and sufficient conditions for a $(|U|+|V|-1)$-element subset $B \subseteq E$ to be a basis. These will serve as a starting point for our analysis of how results from the network simplex algorithm can be transferred to or used for the HAP.

We begin with a simple observation about the structure of the edges in a basis.

Lemma 7.2.1. *Let $G = (U, V, E)$ be a bipartite hypergraph and*

$$E_1 := \{\{u, v\} : u \in U, v \in V\} \subseteq E$$

be the set of all the edges in E. If $B \subseteq E$ is a basis, then $G' = (U, V, B \cap E_1)$ is a forest and has $|B \setminus E_1| + 1$ connected components.

Proof. If G' would not be a forest, there would exist some cycle

$$(v_0, e_1, v_1, \ldots, e_{2n}, v_{2n}),$$

with $v_0 = v_{2n}$ in the bipartite graph G'. Then $\sum_{i=1}^{2n} (-1)^i \cdot (A(G))_{\cdot e_i} = 0$ since the entry for each v_i has exactly two non-zero summands, that sum up to zero—a $(-1)^i$ and a $(-1)^{i+1}$.

To prove the number of connected components, observe that $|U|+|V|-1 = |B| = |B \cap E_1| + |B \setminus E_1|$ implies $|B \cap E_1| = |U| + |V| - (|B \setminus E_1| + 1)$, which is exactly the equation that has to hold for a forest with $|B \setminus E_1| + 1$ connected components. □

$G' = (U, V, B)$ being a tree is a sufficient condition for a $(|U| + |V| - 1)$-element subset of edges $B \subseteq E_1$ to be a basis, which is known from the network simplex algorithm. To show this, let v_0 be some leaf of the tree and let $b \in \mathbb{R}^{(U \uplus V) \setminus \{v_0\}}$ be a vector with $(|U| + |V| - 1)$ entries. Unique values x_e for all the edges such that $b_v = A_{v\cdot} x = \sum_{e \in \delta(v)} x_e$ for all $v \in V$ can be found by repeating the following procedure $(|U|+|V|-1)$ times. Chose a leaf v different than v_0, and let e be the only edge left that is incident to v. Assign the value $b_v - \sum_{e' \in \delta(v) \setminus \{e\}} x_{e'}$ (by construction, all $x_{e'}$ are determined already) to x_e and remove v and e from the tree.

To get an understanding of the structure of the proper hyperedges in B, we will stick to a more structured bipartite hypergraph, a partitioned hypergraph

with maximum part size two. The next lemma shows that the proper hyperedges in a basis for such a hypergraph also form a kind of forest. It can be viewed as a forest consisting of the edges which connect parts in our visualization of hyperedges in partitioned hypergraphs that connect all vertices from two parts, see, for example, Figure 1.4. The connected components of this forest have to be connected by edges in the basis.

Lemma 7.2.2. *Let $G = (U, V, E)$ be a partitioned hypergraph with maximum part size two, and let $U_1, \dots, U_p$ and $V_1, \dots, V_q$ be the parts of size two on the U- and V-side, respectively. If $B \subseteq E$ is a basis, then the bipartite graph $G'' = (U'', V'', E'')$ with*

$$\begin{aligned} U'' &= \{U_i : \ i \in \{1, \dots, p\}\}, \\ V'' &= \left\{V_j : \ j \in \{1, \dots, q\}\right\}, \\ E'' &= \left\{\{U_i, V_j\} : \ U_i \cup V_j \in B\right\}. \end{aligned}$$

is a forest.

Proof. Analogously to the proof of Lemma 7.2.1, if some edges $\{U_i, V_j\}$ form a cycle in G'', the alternating sum of the corresponding columns of the hyperedges $U_i \cup V_j$ in $A(G)$ is 0, which implies that they are linearly dependent. □

Lemma 7.2.1 shows that the subgraph with all vertices from the bipartite hypergraph and the edges of a basis has to be a forest. If this forest is a tree, the network simplex algorithm [Orlin, 1997] allows us to combinatorially perform a step of the simplex algorithm. If proper hyperedges are present in the basis and therefore the edges are not a tree, we present a combinatorial possibility to perform the simplex algorithm step that works for some of the bases, namely, if it is possible to transform the basis to a basis of what we call the tree type. An algorithm that achieves this in some cases can be found in Section 7.3. The simplex step can then be mainly done on the tree basis and then easily transformed back. This will be discussed in Section 7.4. We will also show how for each hyperassignment such a basis can be found that has this hyperassignment as the associated solution. This can be done for an arbitrary bipartite hypergraph, as we will show in Section 7.5.

7.3 Transforming to Bases of the Tree Type

This section deals with the following question: Which bases can be transformed to a basis type that consists of only edges such that a simplex method step in

the edge basis can be transformed back easily? By Lemma 7.2.1, these edges are a spanning tree as are the edges in the basis of, e. g., minimum cost flow problems in graphs, so that we can make use of the network simplex method.

Let $G = (U, V, E)$ be a partitioned hypergraph with maximum part size two and all edges $\{\{u, v\} : u \in U, v \in V\} \subseteq E$ in its hyperedge set. Let $A := A(G)$ be its incidence matrix. Let $F \subseteq E$ be some hyperedge set of cardinality $|F| = |U| + |V| - 1$.

Our method applies elementary column operations to the matrix $A_{\cdot F}$ to transform it to a matrix in which we can check the linear independence of the columns combinatorially. The matrix will look like the coefficient matrix of an assignment problem. In an elementary column operation, a column is replaced by some non-zero multiple of itself plus multiples of other columns. Elementary column operations can be represented by the multiplication from the right side with an upper triangular matrix P. This will allow us to do steps of the simplex algorithm in a combinatorial way by applying the network simplex algorithm and then transforming the result using these elementary column operations as we will show in Section 7.4.

First, we split F into F_1 and F_2 such that $F = F_1 \dot\cup F_2$ and F_1 contains only edges, F_2 only proper hyperedges. From Lemma 7.2.1, we know that the graph with vertex set $U \dot\cup V$ and edge set F_1 must be a forest for the edges in F_1 to be linearly independent. This is also equivalent to being composed of $|U| + |V| - |F_1| = |F_2| + 1$ connected components. Further, every other edge that consists of vertices from just one connected component, is linearly dependent to the edges in this connected component since each connected component is a tree and therefore the edges form a basis for it. For some connected component G' and an edge e using only vertices from G' (but not necessarily an edge that is in the edge set of G'), we can therefore construct an elementary column operation that subtracts $A_{\cdot e}$ from some arbitrary column $A_{\cdot e'}$ for a proper hyperedge e' with $e \subset e'$ in $A_{\cdot F}$. If we compare the corresponding hypergraph for the resulting incidence matrix to the corresponding hypergraph for the original incidence matrix, we can see that it has one hyperedge with two vertices less. The rest stays the same.

We can perform a sequence of such *edge-reduction steps* until no more hyperedges in the basis are left that can be simplified in such a way. A matrix where such an operation cannot be done anymore will be called an *edge-irreducible matrix*. Algorithm 7.3.1 shows how such a search can be performed. If after some edge-reduction step the edges of the resulting hypergraph form a cycle, the algorithm stops and returns that the input hypergraph cannot be a basis. This is correct since elementary column operations do not alter linear independence. If the resulting edge-irreducible matrix corresponds to a tree, we call F

a *tree-transformable basis*.

We will now show how an elementary column operation works in more detail. An elementary column operation that adds m times the column e' to the column e, $e \neq e'$ can be represented by the matrix $P(e,e',m) = (p_{f,g})_{f,g\in E} \in \mathbb{R}^E$ defined by

$$p_{f,g} := \begin{cases} 1 & \text{if } f = g \\ m & \text{if } f = e', g = e \\ 0 & \text{otherwise.} \end{cases}$$

Then, $A_{\cdot e'} = m_1 \cdot A_{\cdot e_1} + m_2 \cdot A_{\cdot e_2} + \ldots + m_k \cdot A_{\cdot e_k}$ for some real numbers $m_1, \ldots, m_k$ implies that in matrix

$$A_{\cdot F} \cdot P(e, e_1, -m_1) \cdot P(e, e_2, -m_2) \cdots P(e, e_k, -m_k)$$

all columns besides the one for e are the same as in $A_{\cdot F}$, and the one for e is $A_{\cdot e} - A_{\cdot e'}$.

$P(e,e',m)$ is invertible, and the inverse is $P(e,e',-m)$.

7.4 Performing a Simplex Step after Transformation to a Tree Basis

We have seen in the previous section how certain bases $B \subseteq E$ for bipartite hypergraphs $G = (U,V,E)$ with edge set $E_1 = \{\{u,v\} : u \in U, v \in V\} \subseteq E$ in the hyperedge set can be transformed to a basis $B' \subseteq E_1$ consisting only of edges such that

$$(A(G))_{\cdot B'} = (A(G))_{\cdot B} \cdot P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k)$$

for some elementary column operation matrices $P(e_i, f_i, m_i)$.

Assume that a cost function $c_E : E \to \mathbb{R}$ is given. We denote for a set $X \subseteq E$ by $c_X \in \mathbb{R}^X$ the vector with entries $(c_X)_e = c_E(e)$. Further, chose the matrix A such that it results from $A(G)$ by deletion of the row for some vertex v_0, in other words,

$$A := (A(G))_{(U\cup V)\setminus\{v_0\}\ E}.$$

For the graph $G' = (U,V,E_1)$ and a tree basis $B' \subseteq E_1$, the network simplex algorithm [Orlin, 1997] allows us to compute $A_{\cdot B'}^{-1} \cdot A_{\cdot e}$ for all edges $e = \{u,v\} \in E_1 \setminus B'$ combinatorially. This is done by finding the unique $[u,v]$-path $(u, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v)$ in the spanning tree (U,V,B'). Then,

$$(A_{\cdot B'}^{-1} \cdot A_{\cdot e})_{e'} = \begin{cases} (-1)^{k+1} & \text{if } e' = e_k \text{ for some } k \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 7.3.1: Computes for a hypergraph $G = (U, V, E)$ an edge-irreducible hypergraph $G' = (U, V, E')$ and a list of elementary column operations that transform $A(G)$ to $A(G')$, or finds that $A(G)$ is not a basis.

Data: Bipartite hypergraph $G = (U, V, E)$.
Result: Edge-irreducible bipartite hypergraph $G' = (U, V, E')$ and an ordered list $\mathsf{L} = ((e_1, f_1, m_1), \ldots, (e_k, f_k, m_k))$ such that $A(G') = A(G) \cdot P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k)$, or finds that the columns of $A(G)$ are linearly dependent.

```
1  G' = (U, V, E') ← G                                   // initialize E' with E
2  L ← empty list
3  repeat
4      NewEdgeAdded ← false
5      E_1 ← {e ∈ E' : |e| = 2}                                  // edges of G'
6      E_2 ← E' \ E_1                                  // proper hyperedges of G'
7      G'_1 = (U'_1 ∪ V'_1, E'_1), ..., G'_n = (U'_n ∪ V'_n, E'_n) ← connected components of
       (U ∪ V, E_1) with U'_i ⊆ U, V'_i ⊆ V
8      if G'_i is not a tree for some i ∈ {1, ..., n} then
9          return A(G) has linearly dependent columns
10     foreach e ∈ E_2 do
11         for i ← 1 to n do
12             if |e ∩ U'_i| > 0 and |e ∩ V'_i| > 0 then
13                 u ← a vertex from e ∩ U'_i
14                 v ← a vertex from e ∩ V'_i
15                 e ← e \ {u, v}                          // remove u and v from e
16                 e'_1, ... e'_m ← unique [u, v]-path in G'_i
17                 for j ← 1 to m do append (e, e'_j, (−1)^j) to L          // save
                   elementary column operation in L
18                 if |e|=2 then                              // if e is now an edge
19                     if e ⊆ U'_j ∪ V'_j for some j ∈ {1, ..., n} then
20                         return A(G) has linearly dependent columns
21                     else
22                         NewEdgeAdded ← true
23                         break
24         if NewEdgeAdded then
25             break // connected components have changed, we have
               to redefine them
26 until not NewEdgeAdded// try again if new edges were added and
   thus connected components were changed
27 return (G', L)
```

Further, the reduced costs $\bar{c}(e) = c_E(e) - c_{B'}^T A_{.B'}^{-1} \cdot A_{.e}$ for $e \in E_1$ can be computed combinatorially in the graph case. More specifically, the entries π_v of the vector $c_{B'}^T A_{.B'}^{-1} \in \mathbb{R}^{(U \cup V) \setminus v_0}$ can be found by simple operations in the tree (U, V, B'). Define $\pi_{v_0} := 0$. Then, $\bar{c}(e) = c_E(e) - \pi_u - \pi_v$ for $e = (u, v) \in E_1$. To get the values π_v for $v \in (U \cup V) \setminus \{v_0\}$, observe that $\pi = c_{B'}^T A_{.B'}^{-1}$ implies that $c_{B'}^T = \pi A_{.B'}^{-1}$ and therefore $\pi_v = \sum_{e \in \delta(v)} c_E(e)$. If the tree (U, V, B') is a path starting at v_0, i. e., there exists a path

$$(v_0, e_1, v_1, e_2, \ldots, v_{|B'|-1}, e_{|B'|}, v_{|B'|})$$

that consists of all vertices and edges of the tree—this is the case that we will employ later—the values of π_v can thus be determined by

$$\pi_{v_i} = c_E(\{v_i, v_{i-1}\}) - \pi_{v_{i-1}}$$

for first $i = 1$, then $i = 2$, …, and finally $i = |B'|$.

As discussed in the last paragraph, $A_{.B'}^{-1} \cdot A_{.e}$ for $e \in E_1$ and $c_{B'}^T \cdot A_{.B'}^{-1}$ can be computed easily by combinatorial procedures. We will now show how this can be used to get also $A_{.B}^{-1} \cdot A_{.e}$ for $e \in E_1$, then $A_{.B}^{-1} \cdot A_{.e}$ for $e \in E$, and lastly $c_B^T \cdot A_{.B}^{-1}$, which implies the reduced costs.

- To compute $A_{.B}^{-1} \cdot A_{.e}$ for $e \in E_1$, observe that

  $$A_{.B}^{-1} = P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k) \cdot A_{.B'}^{-1}.$$

  Therefore, $A_{.B}^{-1} \cdot A_{.e}$ results from $A_{.B'}^{-1} \cdot A_{.e}$ by multiplication with the elementary column operation matrices from the left. These multiplications are just additions of multiples of entries.

- From $A_{.B}^{-1} \cdot A_{.e}$ for $e \in E_1$ we can get $A_{.B}^{-1} \cdot A_{.e}$ also for $e \in E \setminus E_1$. Since for all $e \in E \setminus E_1$ there exist $e_1, e_2, \ldots e_k \in E_1$ such that $e = e_1 \cup e_2 \cup \ldots \cup e_k$, we can compute $A_{.B}^{-1} \cdot A_{.e}$ by adding $A_{.B}^{-1} \cdot A_{.e_1}, A_{.B}^{-1} \cdot A_{.e_2}, \ldots, A_{.B}^{-1} \cdot A_{.e_k}$.

- For $c_B^T \cdot A_{.B}^{-1}$, we can again substitute $A_{.B}^{-1}$ by $P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k) \cdot A_{.B'}^{-1}$ to get $c_B^T \cdot P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k) \cdot A_{.B'}^{-1}$. This means that we can first alter c_B^T by the elementary column operations to get some $c_B^{'T}$, and then compute the entries π_v of $c_B^{'T} \cdot A_{.B'}^{-1}$ combinatorially as in the network simplex algorithm. Again, let $\pi_{v_0} = 0$. The reduced costs $\bar{c}(e) = c_E(e) - c_B^T A_{.B}^{-1} \cdot A_{.e}$ can then be computed as

  $$\bar{c}(e) = c_E(e) - \sum_{u \in e \cap U} \pi_u - \sum_{v \in e \cap V} \pi_v$$

  for $e \in E$.

7.5 Finding a Tree-Transformable Basis for a Hyperassignment

In this section, we show how the set of hyperedges in a hyperassignment can be extended to a tree-transformable basis, i. e., such that the basis can be transformed by Algorithm 7.3.1 to a basis of the tree type.

Let H be a hyperassignment in some bipartite hypergraph $G = (U, V, E)$ with $E_1 = \{\{u, v\} : u \in U, v \in V\} \subseteq E$.

For each proper hyperedge $e \in H \setminus E_1$, let $E_1(e)$ be some set of pairwise disjoint edges $e_1, e_2, \ldots, e_k$ such that their union $\bigcup E_1(e)$ is equal to e. Besides the hyperedges in H, add for each proper hyperedge e, the edges $e_2, \ldots, e_k \in E_1(e)$ to the basis, i. e., all but one of the edges that have the proper hyperedge as its union. Then, Algorithm 7.3.1 can transform e to e_1, and no proper hyperedges are left. After the transformation, we have

$$\{e \in H : e \in E_1\} \cup \bigcup_{e \in H \setminus E_1} E_1(e)$$

in the basis. We can now complete the basis by adding edges such that the result is a spanning tree (which is also the right number of elements in a basis). A simple tree spanning tree to use would be a path.

This method is summarized in Algorithm 7.5.1. It constructs the basis and returns also the tree basis and the elementary column operations that transform the basis to it. The operations that could be found by Algorithm 7.3.1 are also included since they are directly implied by the construction.

7.6 A Primal Combinatorial Algorithm for the HAP

Algorithm 7.6.1 shows in detail how the very large-scale neighborhood search from Section 7.1 can be combined with the composite columns method to escape a local minimum using the methods developed in Sections 7.2–7.5. In each round of the while loop, a better hyperassignment is found if it exists. Either, a local search step is performed, or, if this is not possible and hence a local minimum is reached, a tree-transformable basis is found for the current solution as described in Section 7.5. This basis is then used to compute the input for the composite columns method for the most part by combinatorial operations from the network simplex algorithm. If a better integral solution exists (and therefore is found by the composite columns method), we switch back to the local search and repeat the procedure.

Algorithm 7.5.1: Finds a tree-transformable basis for a hyperassignment.

Data: Bipartite hypergraph $G = (U, V, E)$ with all edges $E_1 = \{\{u, v\} : u \in U, v \in V\} \subseteq E$ in its edge set, and hyperassignment $H \subseteq E$.

Result: Basis $\mathsf{B} \subseteq E$ such that $H \subseteq \mathsf{B}$, basis $\mathsf{B}' \subseteq E_1$, an ordered list $L = ((e_1, f_1, m_1), \ldots, (e_k, f_k, m_k))$ such that $A(G)_{\cdot \mathsf{B}'} = A(G)_{\cdot \mathsf{B}} \cdot P(e_1, f_1, m_1) \cdots P(e_k, f_k, m_k)$.

```
1  B ← empty list
2  B' ← empty list
3  L ← empty list
4  foreach e ∈ H do
5      if e ∈ E_1 then                                  // if e is an edge
6          append e to B
7          append e to B'
8      else                                  // if e is a proper hyperedge
9          U' ← e ∩ U
10         V' ← e ∩ V
           //  e = {U'[1], V'[1]} ∪ {U'[2], V'[2]} ∪ ... ∪ {U'[|U'|], V'[|U'|]}
11         append e to B
12         append {U'[1], V'[1]} to B'
13         for i ← 2 to |U'| do
14             e' ← {U'[i], V'[i]}
15             append e' to B
16             append e' to B'
17             append (e, e', −1) to L                  // transform proper
                   hyperedges to edges
18 N ← |B'|
19 for i ← 1 to N − 1 do     // connect the edges in B' to a path
20     e ← (B'[i] ∩ U) ∪ (B'[i+1] ∩ V)
21     append e to B
22     append e to B'
23 return (B, B', L)
```

Algorithm 7.6.1: Local Search with Network CoCo.

Data: Partitioned hypergraph $G = (U, V, E)$ such that $E_1 := \{\{u, v\} : u \in U, v \in V\} \subseteq E$.
Result: Optimal hyperassignment $H \subseteq E$ in G w. r. t. c_E.

$\mathcal{G} \leftarrow \{\{v\} : v \in U \cup V\}$ `// finest vertex grouping`
$H \leftarrow$ hyperassignment in G with minimum cost w. r. t. c_E that respects $\mathcal{G}$
`// use, e.g., Hungarian algorithm`
Direction $\leftarrow 1$ `// start local search in coarsening direction`
while true do `// while optimal solution not found and returned`
  BestNeighbor $\leftarrow$ output of Algorithm 7.1.1 with input G, c_E, H, Direction
  `// try local search step`
  if c_E(BestNeighbor) $< c_E(H)$ **then** `// better solution found`
    break `// continue with loc. s. in the same direction`
  else
    Direction $\leftarrow 1 -$ Direction `// switch direction`
    BestNeighbor $\leftarrow$ output of Algorithm 7.1.1 with input G, c_E, H, Direction `// try local search step again in the other direction`
    if c_E(BestNeighbor) $< c_E(H)$ **then** `// better solution found`
      break `// continue with loc. s. in the new direction`
  `// local search is in a local minimum, continue with composite columns`
  foreach $e \in E$ **do**
    if $e \in H$ **then** CurrentSolution[e]=1 **else** CurrentSolution[e]=0
  (B, B', L) $\leftarrow$ output of Algorithm 7.5.1 `// find tree-transformable basis, tree basis and list with elementary column operations for transformation`
  compute simplex tableau columns $A_{\cdot B}^{-1} \cdot A_{\cdot e}$ and reduced costs $\bar{c}(e)$ for all $e \in E \setminus H$ using the network simplex algorithm and B', L as described in Section 7.4
  apply Algorithm I from [Balas and Padberg, 1975] with the calculated simplex tableau, reduced costs and current solution CurrentSolution until an integral solution with lower cost is found or the Algorithm finds that it does not exist
  if *solution x with lower cost exists* **then**
    $H \leftarrow \{e \in E : x_e = 1\}$
    break `// continue with local search`
  else
    return H `// H is an optimal solution`

Appendix A

Data Tables for Random HAP with Regularity Rewarding Costs

The following table shows computational results for random hypergraph assignment problems in $G_{2,n}$ with regularity rewarding costs. For an explanation, see Section 4.2. The mean optimal values (column 3) and its standard deviations (column 4) are rounded to the third decimal place, its relative standard deviations (column 5) to the second decimal place. The numbers of proper hyperedges in the found optimal hyperassignment (column 6) and their standard deviations (column 8) are rounded to two decimal places. The relative numbers of proper hyperedges in the found optimal hyperassignment (column 7) and their standard deviations (column 9) are rounded to one decimal place.

p	n	o. v. mean	o. v. s. d.	o. v. r. s. d.	# p. h. mean	r. # p. h. mean	# p. h. s. d.	r. # p. h. s. d.
0	30	1.555	0.160	10.30 %	0.02	0.1 %	0.14	0.5 %
0	60	1.648	0.129	7.84 %	0.03	0.1 %	0.17	0.3 %
0	90	1.623	0.089	5.46 %	0.02	0.0 %	0.14	0.2 %
0	120	1.635	0.098	6.01 %	0.03	0.0 %	0.17	0.1 %
0	150	1.636	0.078	4.76 %	0.05	0.0 %	0.22	0.1 %
0	180	1.629	0.073	4.51 %	0.02	0.0 %	0.14	0.1 %
0	210	1.643	0.061	3.72 %	0.03	0.0 %	0.17	0.1 %
0	240	1.640	0.058	3.55 %	0.01	0.0 %	0.10	0.0 %
0	270	1.638	0.055	3.37 %	0.03	0.0 %	0.17	0.1 %
0	300	1.639	0.058	3.54 %	0.04	0.0 %	0.20	0.1 %
0.01	30	2.178	0.157	7.22 %	1.39	4.6 %	1.22	4.1 %
0.01	60	2.797	0.132	4.72 %	2.76	4.6 %	1.56	2.6 %
0.01	90	3.378	0.086	2.54 %	4.76	5.3 %	1.98	2.2 %

Table A.1 continues on the next page

Table A.1, continued from the previous page

p	n	o. v. mean	o. v. s. d.	o. v. r. s. d.	# p. h. mean	r. # p. h. mean	# p. h. s. d.	r. # p. h. s. d.
0.01	120	3.970	0.084	2.11 %	7.4	6.2 %	2.45	2.0 %
0.01	150	4.546	0.078	1.71 %	10.97	7.3 %	3.18	2.1 %
0.01	180	5.128	0.085	1.65 %	13.72	7.6 %	3.55	2.0 %
0.01	210	5.696	0.072	1.26 %	18.22	8.7 %	4.25	2.0 %
0.01	240	6.245	0.073	1.17 %	23.33	9.7 %	4.09	1.7 %
0.01	270	6.814	0.061	0.90 %	29.25	10.8 %	5.80	2.1 %
0.01	300	7.373	0.076	1.03 %	34.12	11.4 %	5.28	1.8 %
0.02	30	2.731	0.176	6.46 %	2.71	9.0 %	1.83	6.1 %
0.02	60	3.918	0.128	3.27 %	6.69	11.2 %	2.67	4.5 %
0.02	90	5.022	0.121	2.41 %	12.93	14.4 %	3.02	3.4 %
0.02	120	6.086	0.105	1.72 %	22.17	18.5 %	3.67	3.1 %
0.02	150	7.151	0.125	1.75 %	31.49	21.0 %	4.45	3.0 %
0.02	180	8.163	0.114	1.40 %	43.05	23.9 %	5.20	2.9 %
0.02	210	9.182	0.143	1.56 %	56.01	26.7 %	6.82	3.2 %
0.02	240	10.150	0.144	1.42 %	70.44	29.4 %	6.69	2.8 %
0.02	270	11.092	0.161	1.45 %	86.54	32.1 %	7.01	2.6 %
0.02	300	12.028	0.144	1.20 %	103.49	34.5 %	6.72	2.2 %
0.03	30	3.241	0.159	4.92 %	4.57	15.2 %	1.75	5.8 %
0.03	60	4.902	0.149	3.03 %	13.46	22.4 %	2.81	4.7 %
0.03	90	6.430	0.156	2.43 %	24.64	27.4 %	3.78	4.2 %
0.03	120	7.886	0.168	2.13 %	39.94	33.3 %	4.77	4.0 %
0.03	150	9.294	0.195	2.09 %	56.16	37.4 %	5.83	3.9 %
0.03	180	10.564	0.200	1.89 %	76.69	42.6 %	6.06	3.4 %
0.03	210	11.782	0.193	1.63 %	97.3	46.3 %	6.24	3.0 %
0.03	240	13.050	0.224	1.71 %	120.28	50.1 %	5.85	2.4 %
0.03	270	14.236	0.226	1.59 %	142.14	52.6 %	6.30	2.3 %
0.03	300	15.333	0.283	1.84 %	167.65	55.9 %	8.25	2.8 %
0.04	30	3.732	0.198	5.29 %	6.93	23.1 %	2.47	8.2 %
0.04	60	5.770	0.177	3.07 %	19.43	32.4 %	3.42	5.7 %
0.04	90	7.631	0.205	2.68 %	36.66	40.7 %	4.34	4.8 %
0.04	120	9.340	0.235	2.52 %	57.47	47.9 %	5.28	4.4 %
0.04	150	10.878	0.267	2.45 %	80.17	53.4 %	5.41	3.6 %
0.04	180	12.333	0.249	2.02 %	105.16	58.4 %	6.31	3.5 %
0.04	210	13.761	0.267	1.94 %	131.72	62.7 %	6.39	3.0 %
0.04	240	15.018	0.291	1.94 %	159.17	66.3 %	6.30	2.6 %
0.04	270	16.271	0.284	1.74 %	187.42	69.4 %	6.03	2.2 %
0.04	300	17.469	0.346	1.98 %	216.94	72.3 %	6.39	2.1 %
0.05	30	4.215	0.236	5.59 %	9.39	31.3 %	2.70	9.0 %
0.05	60	6.469	0.221	3.42 %	26.85	44.8 %	3.77	6.3 %
0.05	90	8.557	0.263	3.07 %	48.04	53.4 %	4.30	4.8 %

Table A.1 continues on the next page

Table A.1, continued from the previous page

p	n	o. v. mean	o. v. s. d.	o. v. r. s. d.	# p. h. mean	r. # p. h. mean	# p. h. s. d.	r. # p. h. s. d.
0.05	120	10.382	0.282	2.71 %	73.37	61.1 %	4.91	4.1 %
0.05	150	12.059	0.317	2.63 %	100.56	67.0 %	4.96	3.3 %
0.05	180	13.523	0.328	2.42 %	129.38	71.9 %	4.76	2.6 %
0.05	210	14.981	0.357	2.38 %	160.13	76.3 %	6.32	3.0 %
0.05	240	16.270	0.341	2.10 %	190.56	79.4 %	5.34	2.2 %
0.05	270	17.525	0.415	2.37 %	221.89	82.2 %	5.96	2.2 %
0.05	300	18.704	0.424	2.26 %	253.23	84.4 %	5.82	1.9 %
0.06	30	4.602	0.202	4.38 %	12.09	40.3 %	2.90	9.7 %
0.06	60	7.088	0.262	3.69 %	33.24	55.4 %	3.35	5.6 %
0.06	90	9.324	0.292	3.13 %	58.04	64.5 %	5.12	5.7 %
0.06	120	11.141	0.335	3.01 %	87.07	72.6 %	4.24	3.5 %
0.06	150	12.876	0.388	3.01 %	116.4	77.6 %	4.61	3.1 %
0.06	180	14.325	0.380	2.65 %	148.07	82.3 %	4.55	2.5 %
0.06	210	15.784	0.381	2.41 %	179.87	85.7 %	4.36	2.1 %
0.06	240	17.040	0.385	2.26 %	212.39	88.5 %	4.45	1.9 %
0.06	270	18.250	0.380	2.08 %	244.11	90.4 %	4.97	1.8 %
0.06	300	19.433	0.431	2.22 %	276.96	92.3 %	4.89	1.6 %
0.07	30	4.887	0.297	6.08 %	14.69	49.0 %	2.64	8.8 %
0.07	60	7.597	0.306	4.02 %	38.24	63.7 %	3.72	6.2 %
0.07	90	9.846	0.317	3.22 %	66.43	73.8 %	3.69	4.1 %
0.07	120	11.669	0.412	3.53 %	97.92	81.6 %	4.51	3.8 %
0.07	150	13.448	0.430	3.19 %	129.03	86.0 %	4.59	3.1 %
0.07	180	14.901	0.398	2.67 %	162.49	90.3 %	3.69	2.0 %
0.07	210	16.237	0.483	2.98 %	195.03	92.9 %	4.33	2.1 %
0.07	240	17.472	0.460	2.63 %	227.69	94.9 %	4.09	1.7 %
0.07	270	18.643	0.450	2.41 %	259.69	96.2 %	4.03	1.5 %
0.07	300	19.721	0.498	2.53 %	292.67	97.6 %	4.04	1.3 %
0.08	30	5.138	0.262	5.11 %	16.75	55.8 %	2.75	9.2 %
0.08	60	7.986	0.339	4.25 %	43.55	72.6 %	3.61	6.0 %
0.08	90	10.245	0.389	3.80 %	73.6	81.8 %	3.79	4.2 %
0.08	120	12.080	0.381	3.15 %	106.01	88.3 %	3.58	3.0 %
0.08	150	13.743	0.431	3.14 %	137.74	91.8 %	3.97	2.6 %
0.08	180	15.134	0.469	3.10 %	170.97	95.0 %	3.55	2.0 %
0.08	210	16.374	0.404	2.47 %	204.15	97.2 %	3.46	1.6 %
0.08	240	17.576	0.471	2.68 %	236.25	98.4 %	3.16	1.3 %
0.08	270	18.708	0.470	2.51 %	267.84	99.2 %	2.50	0.9 %
0.08	300	19.690	0.484	2.46 %	298.73	99.6 %	2.15	0.7 %
0.09	30	5.407	0.352	6.51 %	19.45	64.8 %	2.95	9.8 %
0.09	60	8.288	0.387	4.67 %	47.66	79.4 %	3.28	5.5 %
0.09	90	10.439	0.419	4.02 %	80.31	89.2 %	3.06	3.4 %

Table A.1 continues on the next page

Table A.1, continued from the previous page

p	n	o. v. mean	o. v. s. d.	o. v. r. s. d.	# p. h. mean	r. # p. h. mean	# p. h. s. d.	r. # p. h. s. d.
0.09	120	12.359	0.508	4.11 %	111.81	93.2 %	3.53	2.9 %
0.09	150	13.932	0.525	3.77 %	145.26	96.8 %	3.09	2.1 %
0.09	180	15.267	0.441	2.89 %	177.41	98.6 %	2.65	1.5 %
0.09	210	16.532	0.457	2.77 %	208.73	99.4 %	2.13	1.0 %
0.09	240	17.663	0.502	2.84 %	239.62	99.8 %	1.00	0.4 %
0.09	270	18.843	0.528	2.80 %	269.83	99.9 %	0.64	0.2 %
0.09	300	19.684	0.481	2.44 %	299.96	100.0 %	0.40	0.1 %
0.1	30	5.626	0.349	6.20 %	21.48	71.6 %	2.48	8.3 %
0.1	60	8.459	0.359	4.24 %	51.18	85.3 %	2.84	4.7 %
0.1	90	10.731	0.432	4.03 %	83.59	92.9 %	3.13	3.5 %
0.1	120	12.411	0.415	3.35 %	116.83	97.4 %	2.76	2.3 %
0.1	150	13.950	0.502	3.60 %	148.47	99.0 %	2.17	1.4 %
0.1	180	15.295	0.477	3.12 %	179.56	99.8 %	1.11	0.6 %
0.1	210	16.580	0.478	2.88 %	209.9	100.0 %	0.61	0.3 %
0.1	240	17.630	0.509	2.89 %	239.98	100.0 %	0.20	0.1 %
0.1	270	18.819	0.442	2.35 %	269.97	100.0 %	0.30	0.1 %
0.1	300	19.679	0.540	2.74 %	299.98	100.0 %	0.20	0.1 %
0.11	30	5.733	0.360	6.28 %	22.74	75.8 %	2.44	8.1 %
0.11	60	8.565	0.418	4.88 %	54.9	91.5 %	2.77	4.6 %
0.11	90	10.700	0.474	4.43 %	87.37	97.1 %	2.53	2.8 %
0.11	120	12.454	0.507	4.07 %	118.85	99.0 %	1.79	1.5 %
0.11	150	13.994	0.486	3.48 %	149.71	99.8 %	0.92	0.6 %
0.11	180	15.282	0.445	2.91 %	179.95	100.0 %	0.36	0.2 %
0.11	210	16.505	0.474	2.87 %	209.97	100.0 %	0.22	0.1 %
0.11	240	17.706	0.540	3.05 %	240	100.0 %	0.00	0.0 %
0.11	270	18.772	0.426	2.27 %	270	100.0 %	0.00	0.0 %
0.11	300	19.820	0.504	2.54 %	300	100.0 %	0.00	0.0 %
0.12	30	5.917	0.401	6.78 %	24.89	83.0 %	2.45	8.2 %
0.12	60	8.670	0.432	4.98 %	57.19	95.3 %	2.26	3.8 %
0.12	90	10.809	0.489	4.52 %	89.06	99.0 %	1.50	1.7 %
0.12	120	12.422	0.439	3.53 %	119.72	99.8 %	0.93	0.8 %
0.12	150	14.130	0.426	3.02 %	150	100.0 %	0.00	0.0 %
0.12	180	15.312	0.554	3.62 %	180	100.0 %	0.00	0.0 %
0.12	210	16.527	0.464	2.81 %	210	100.0 %	0.00	0.0 %
0.12	240	17.577	0.493	2.81 %	240	100.0 %	0.00	0.0 %
0.12	270	18.741	0.507	2.71 %	270	100.0 %	0.00	0.0 %
0.12	300	19.779	0.495	2.50 %	300	100.0 %	0.00	0.0 %
0.13	30	6.005	0.381	6.35 %	26.24	87.5 %	2.17	7.2 %
0.13	60	8.752	0.428	4.89 %	58.47	97.5 %	2.15	3.6 %
0.13	90	10.841	0.510	4.70 %	89.84	99.8 %	0.77	0.9 %

Table A.1 continues on the next page

Table A.1, continued from the previous page

p	n	o. v. mean	o. v. s. d.	o. v. r. s. d.	# p. h. mean	r. # p. h. mean	# p. h. s. d.	r. # p. h. s. d.
0.13	120	12.435	0.505	4.06 %	119.96	100.0 %	0.28	0.2 %
0.13	150	13.924	0.536	3.85 %	150	100.0 %	0.00	0.0 %
0.13	180	15.251	0.529	3.47 %	180	100.0 %	0.00	0.0 %
0.13	210	16.524	0.496	3.00 %	210	100.0 %	0.00	0.0 %
0.13	240	17.614	0.426	2.42 %	240	100.0 %	0.00	0.0 %
0.13	270	18.750	0.573	3.06 %	270	100.0 %	0.00	0.0 %
0.13	300	19.802	0.492	2.49 %	300	100.0 %	0.00	0.0 %
0.14	30	5.993	0.447	7.45 %	27.95	93.2 %	1.99	6.6 %
0.14	60	8.760	0.478	5.45 %	59.49	99.2 %	1.22	2.0 %
0.14	90	10.832	0.470	4.34 %	89.94	99.9 %	0.45	0.5 %
0.14	120	12.498	0.466	3.73 %	120	100.0 %	0.00	0.0 %
0.14	150	13.906	0.498	3.58 %	150	100.0 %	0.00	0.0 %
0.14	180	15.314	0.508	3.32 %	180	100.0 %	0.00	0.0 %
0.14	210	16.504	0.456	2.76 %	210	100.0 %	0.00	0.0 %
0.14	240	17.658	0.545	3.08 %	240	100.0 %	0.00	0.0 %
0.14	270	18.750	0.474	2.53 %	270	100.0 %	0.00	0.0 %
0.14	300	19.733	0.507	2.57 %	300	100.0 %	0.00	0.0 %
0.15	30	6.080	0.436	7.17 %	28.64	95.5 %	1.71	5.7 %
0.15	60	8.815	0.498	5.65 %	59.85	99.8 %	0.63	1.0 %
0.15	90	10.755	0.471	4.38 %	90	100.0 %	0.00	0.0 %
0.15	120	12.530	0.503	4.01 %	120	100.0 %	0.00	0.0 %
0.15	150	13.891	0.442	3.18 %	150	100.0 %	0.00	0.0 %
0.15	180	15.305	0.460	3.01 %	180	100.0 %	0.00	0.0 %
0.15	210	16.570	0.438	2.64 %	210	100.0 %	0.00	0.0 %
0.15	240	17.723	0.425	2.40 %	240	100.0 %	0.00	0.0 %
0.15	270	18.820	0.515	2.74 %	270	100.0 %	0.00	0.0 %
0.15	300	19.791	0.517	2.61 %	300	100.0 %	0.00	0.0 %

Appendix B

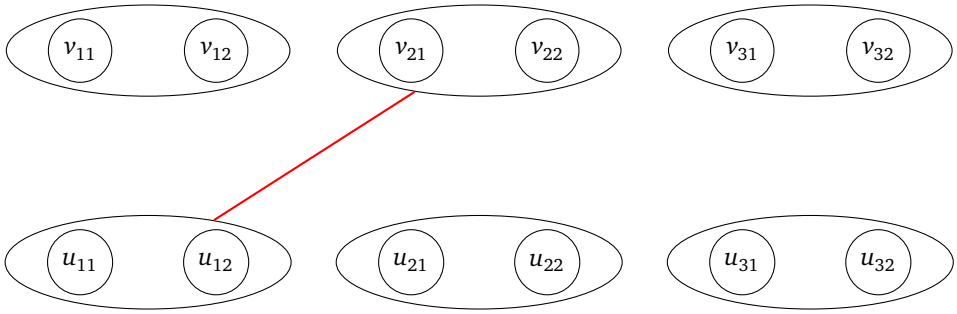
Facets of the HAP Polytope for $G_{2,3}$

We list here one facet from each of the 30 facet classes of $P(\text{HAP})$ for $G_{2,3}$. For an explanation, see Section 5.5.

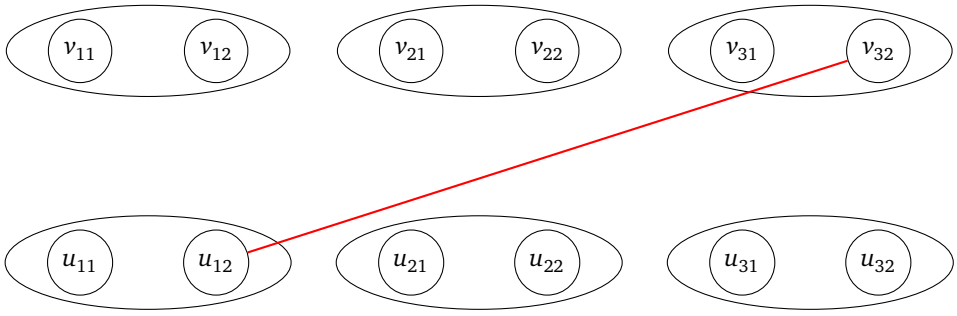
The facets are represented in the form $\sum_{e \in E} a_e x_e \leq b$. The right-hand side b is stated and the coefficients a_e can be seen in the visualization. Hyperedges with coefficient 0 are not drawn. Hyperedges drawn in red have coefficient -1. Hyperedges drawn in black have coefficient 1. Hyperedges drawn in blue have coefficient 2. Except the non-negativity constraints, which are otherwise not easy to recognize, all facets are shown in a representation with only non-negative coefficients. For the understood facets, their types are indicated. For the facets that can be represented as Inequality (OSI_SSP), the least possible $|\mathscr{Q}'|$ for $p = 2$ and $R = 1$ is specified.

B.1 Understood Facet Classes

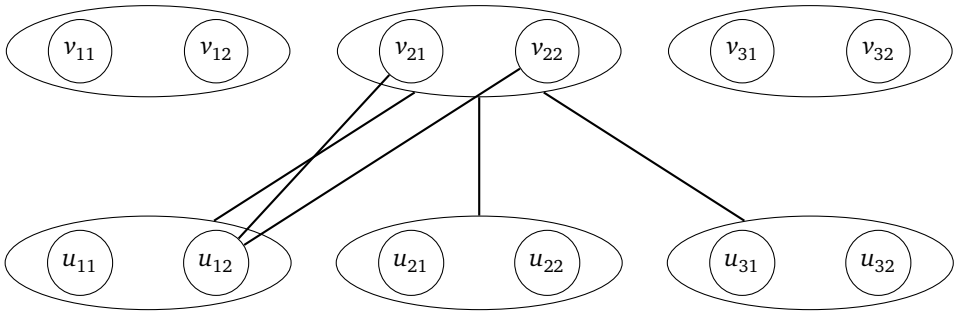
Non-negativity constraint: The following facet class has 9 elements. The right-hand side b is 0.



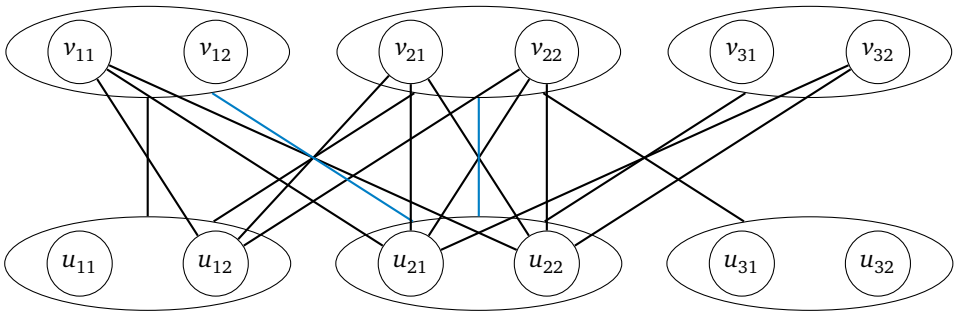
Non-negativity constraint: The following facet class has 36 elements. The right-hand side b is 0.



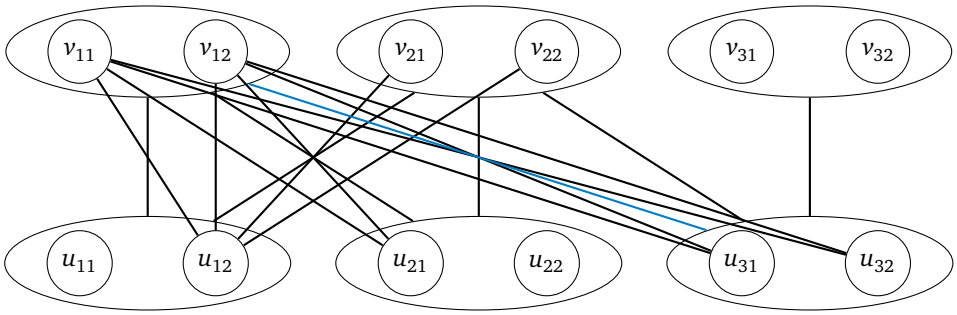
Clique inequality: The following facet class has 36 elements. The right-hand side b is 1.



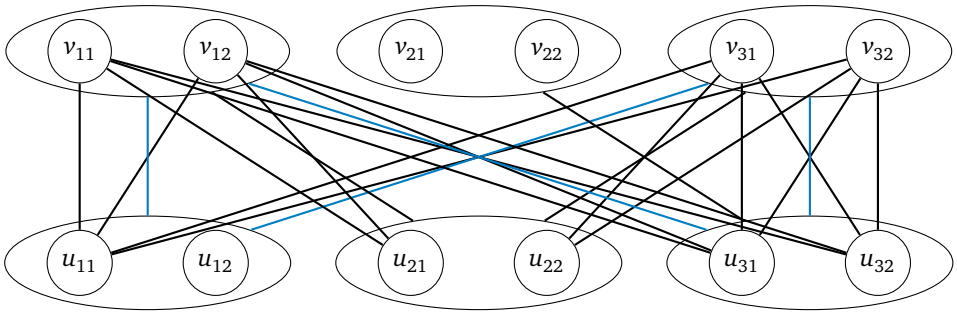
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 7$: The following facet class has 576 elements. The right-hand side b is 3.



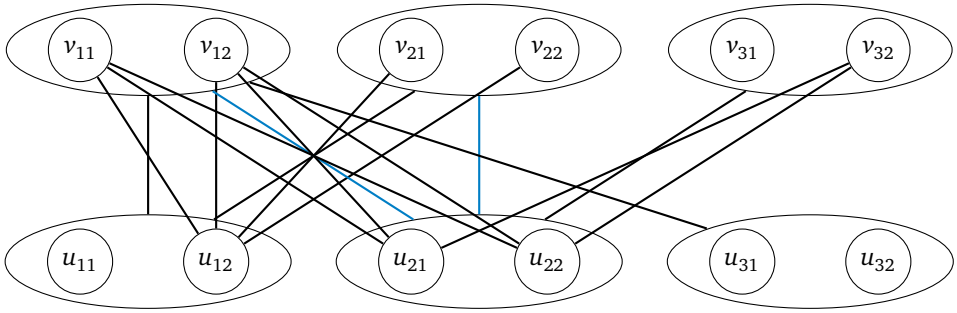
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 7$: The following facet class has 288 elements. The right-hand side b is 3.



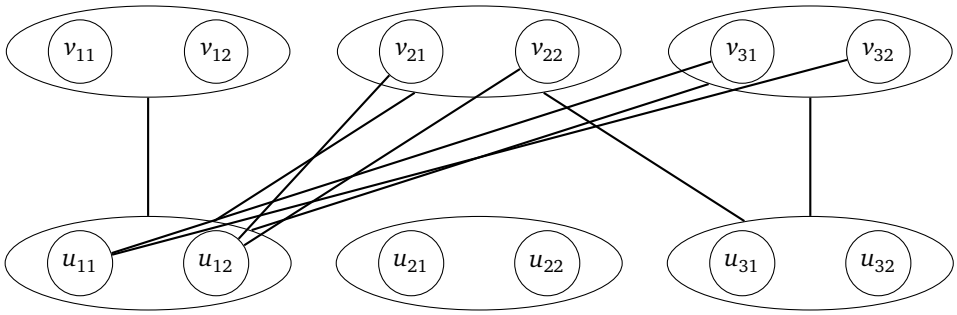
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 7$: The following facet class has 144 elements. The right-hand side b is 4.



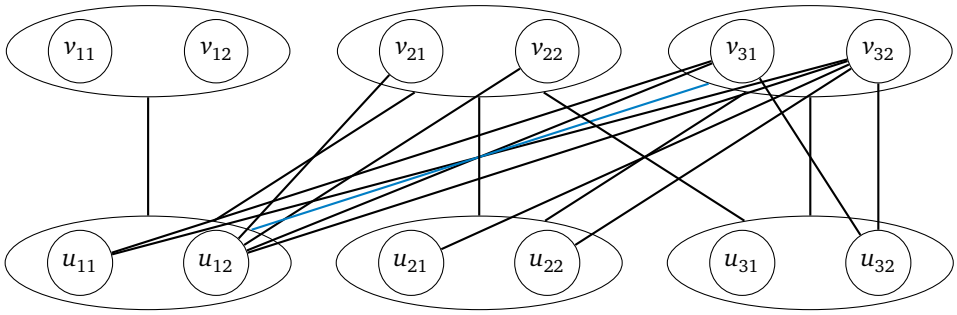
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 7$: The following facet class has 288 elements. The right-hand side b is 3.



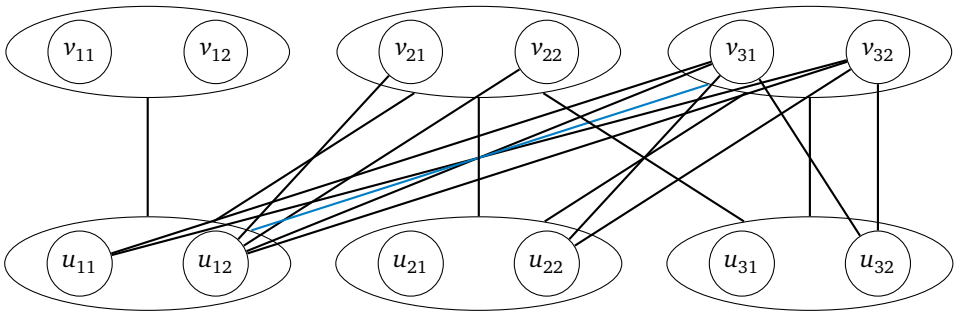
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 5$: The following facet class has 72 elements. The right-hand side b is 2.



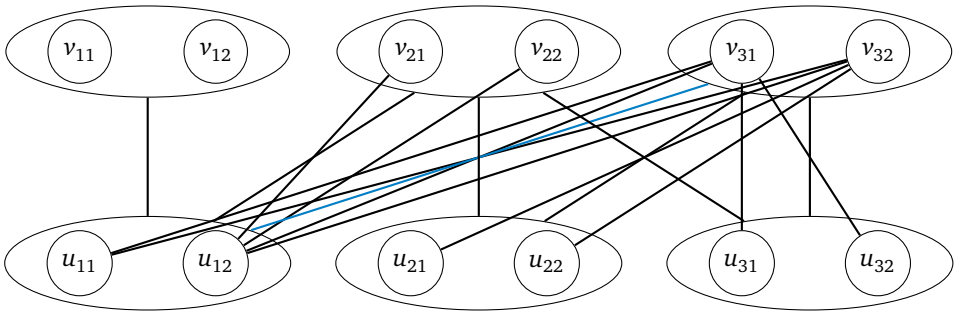
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 7$: The following facet class has 576 elements. The right-hand side b is 3.



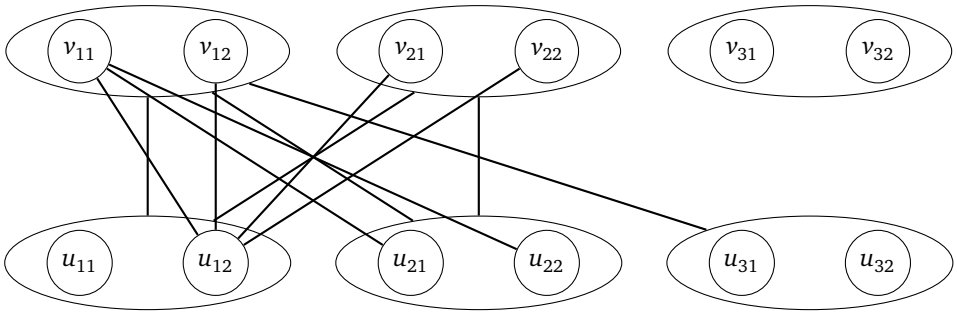
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 7$: The following facet class has 288 elements. The right-hand side b is 3.



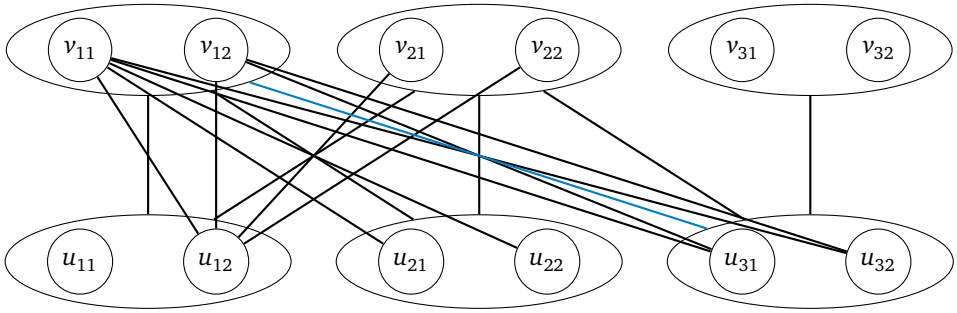
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 7$: The following facet class has 144 elements. The right-hand side b is 3.



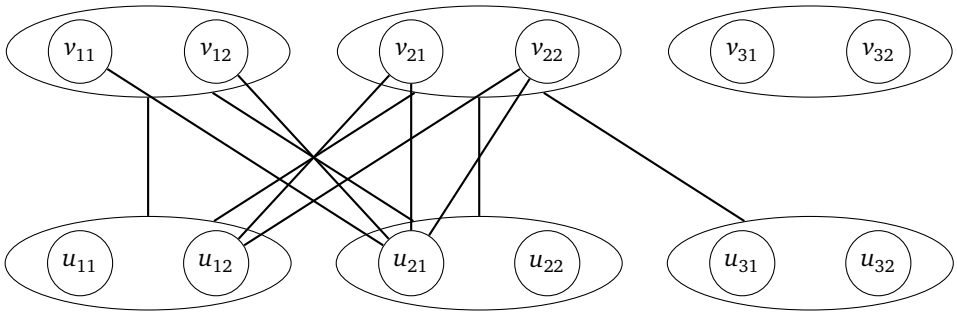
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 5$: The following facet class has 288 elements. The right-hand side b is 2.



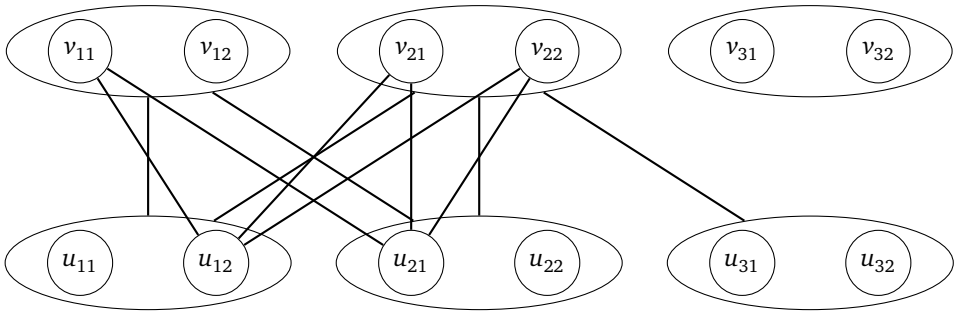
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 7$: The following facet class has 144 elements. The right-hand side b is 3.



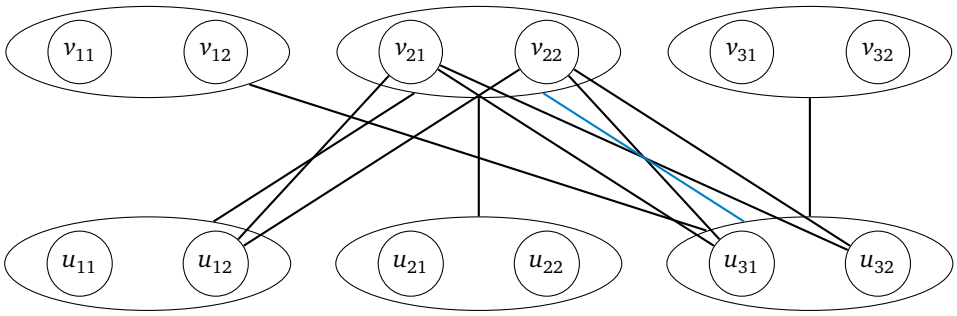
Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 5$: The following facet class has 288 elements. The right-hand side b is 2.



Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathcal{Q}'| = 5$: The following facet class has 288 elements. The right-hand side b is 2.

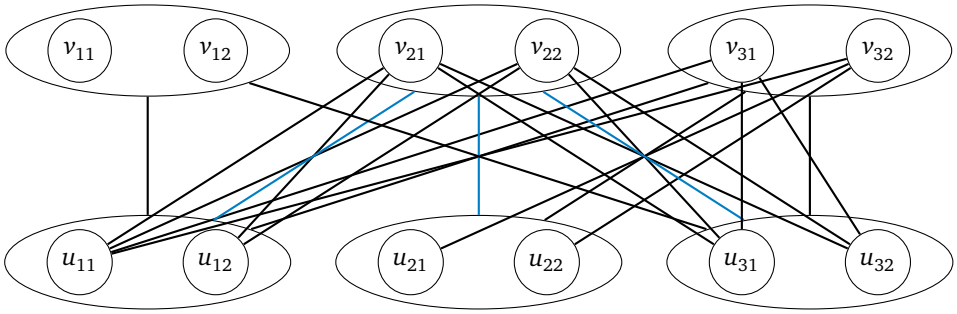


Inequality (OSI_SSP) for $p = 2$, $R = 1$, $|\mathscr{Q}'| = 5$: The following facet class has 72 elements. The right-hand side b is 2.

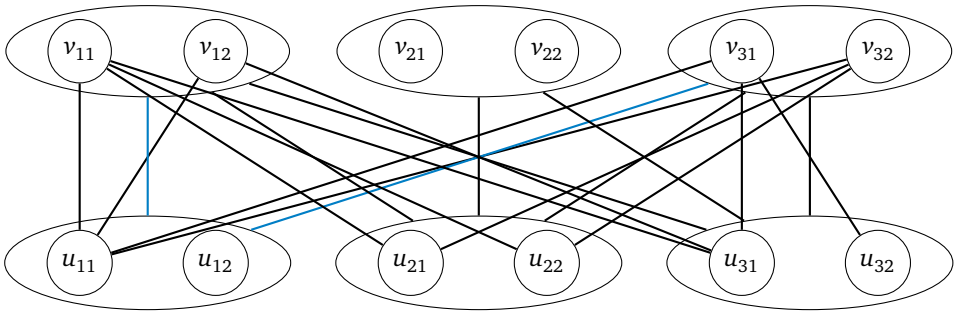


B.2 Other Facet Classes

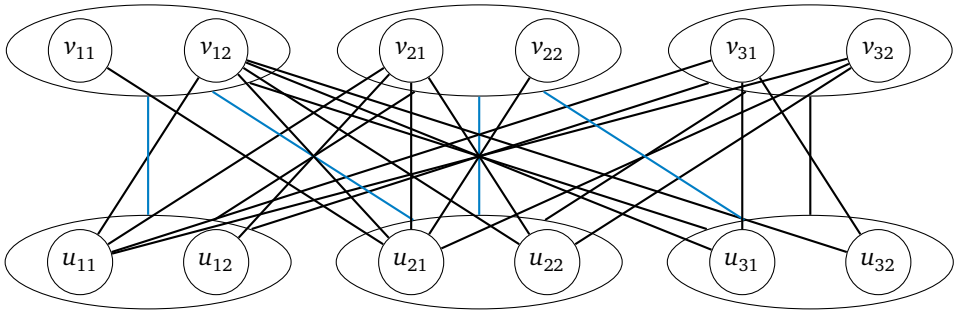
The following facet class has 288 elements. The right-hand side b is 4.



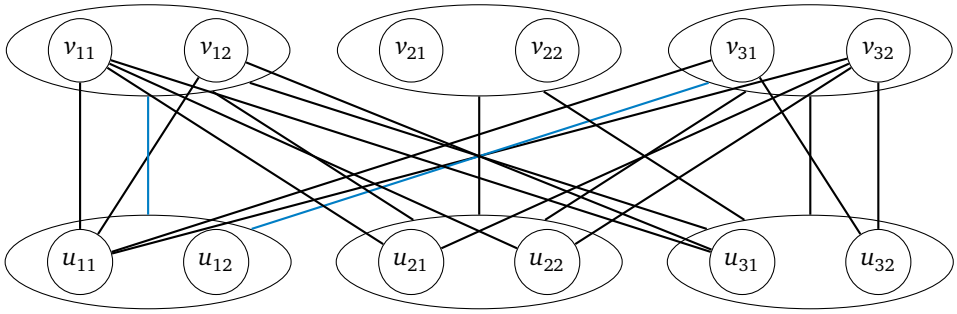
The following facet class has 1152 elements. The right-hand side b is 4.



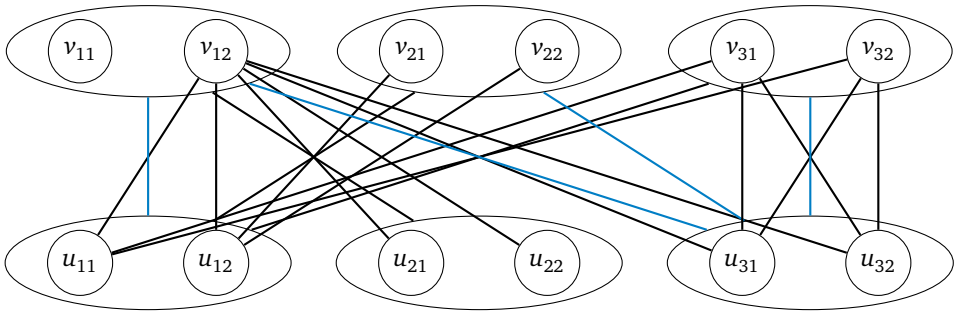
The following facet class has 2304 elements. The right-hand side b is 5.



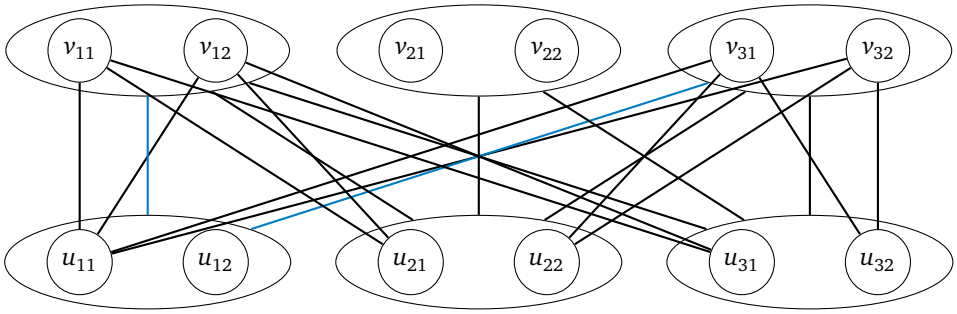
The following facet class has 576 elements. The right-hand side b is 4.



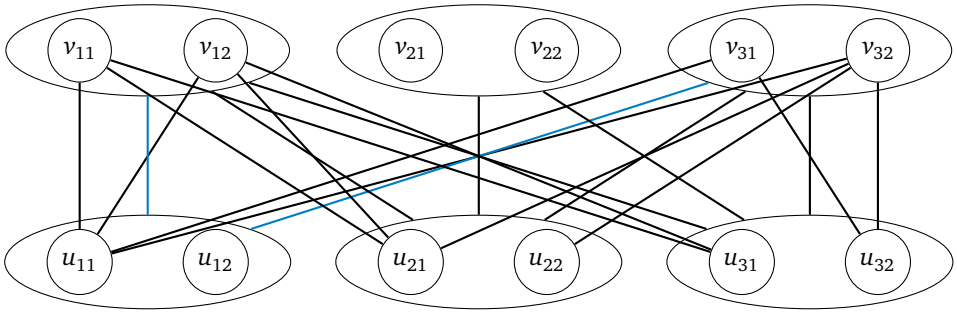
The following facet class has 144 elements. The right-hand side b is 4.



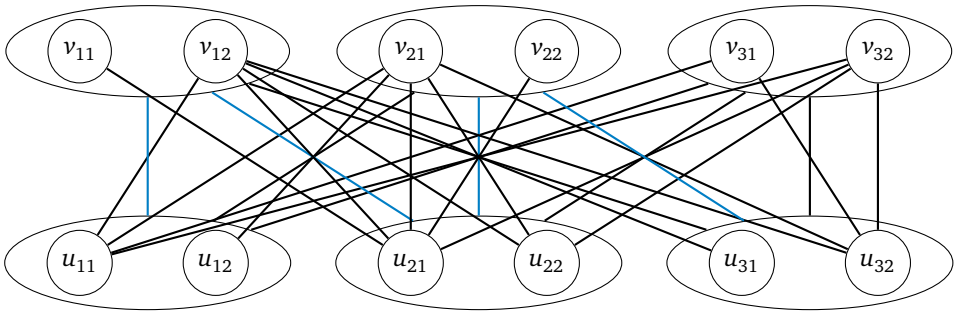
The following facet class has 144 elements. The right-hand side b is 4.



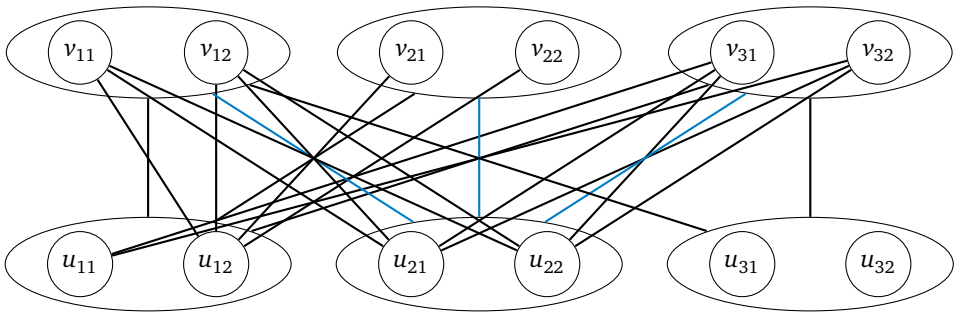
The following facet class has 1152 elements. The right-hand side b is 4.



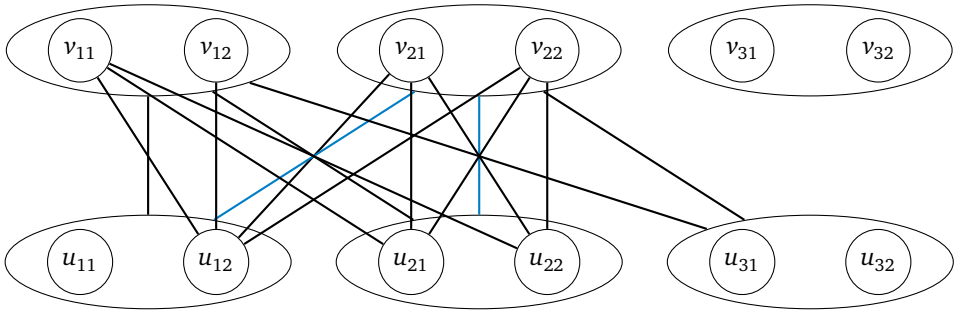
The following facet class has 2304 elements. The right-hand side b is 5.



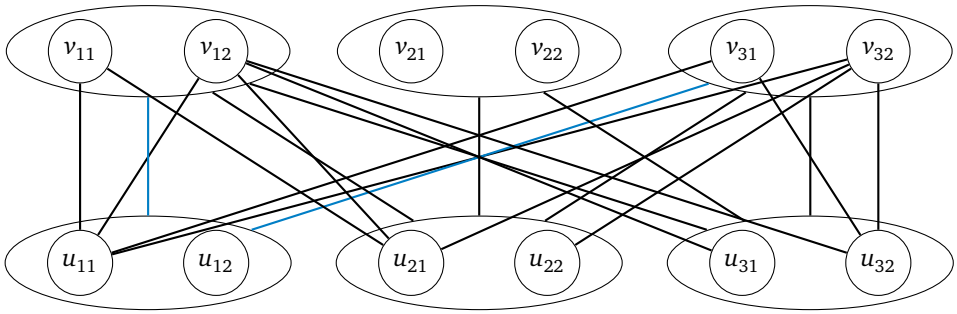
The following facet class has 144 elements. The right-hand side b is 4.



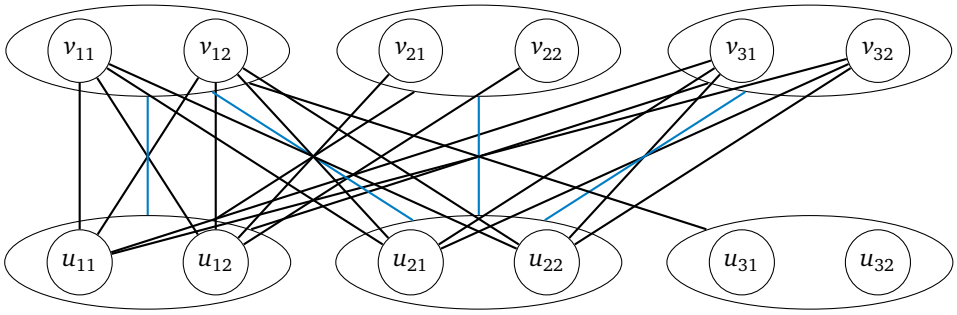
The following facet class has 144 elements. The right-hand side b is 3.



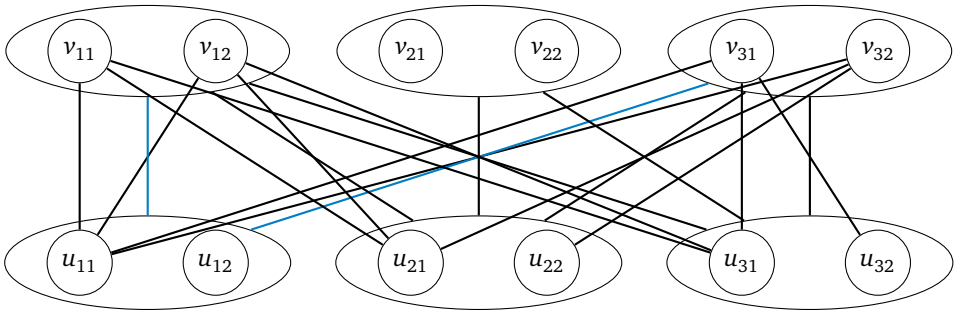
The following facet class has 1152 elements. The right-hand side b is 4.



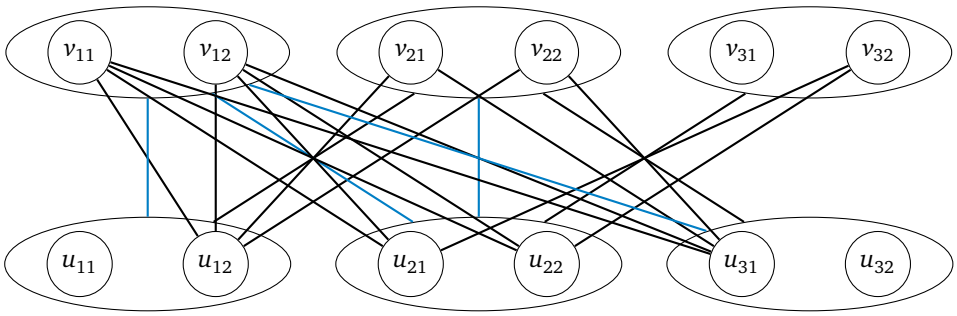
The following facet class has 144 elements. The right-hand side b is 4.



The following facet class has 576 elements. The right-hand side b is 4.



The following facet class has 288 elements. The right-hand side b is 4.



List of Tables

List of Figures

List of Algorithms

Bibliography

Aharoni, R. and Haxell, P. (2000). Hall's theorem for hypergraphs. *Journal of Graph Theory*, 35(2):83–88.

Ahuja, R. K., Ergun, Ö., Orlin, J. B., and Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1–3):75 – 102.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice Hall.

Aldous, D. (1992). Asymptotics in the random assignment problem. *Probability Theory and Related Fields*, 93:507–534.

Amberg, B., Amberg, B., and Kliewer, N. (2011). Approaches for increasing the similarity of resource schedules in public transport. *Procedia – Social and Behavioral Sciences*, 20(0):836–845.

Anstee, R. P. (1987). A polynomial algorithm for b-matchings: An alternative approach. *Information Processing Letters*, 24(3):153–157.

Atamtürk, A., Nemhauser, G. L., and Savelsbergh, M. W. P. (1996). A combined lagrangian, linear programming, and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics*, 1(2):247–259.

Balas, E. (1977). Some valid inequalities for the set partitioning problem. *Annals of Discrete Mathematics*, 1:13–47.

Balas, E. and Carrera, M. C. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890.

Balas, E. and Ng, S. M. (1989a). On the set covering polytope: I. all the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43(1-3):57–69.

Balas, E. and Ng, S. M. (1989b). On the set covering polytope: Ii. lifting the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 45(1-3):1–20.

Balas, E. and Padberg, M. (1975). On the set-covering problem: II. an algorithm for set partitioning. *Operations Research*, 23(1):74–90.

Balas, E. and Padberg, M. W. (1972). On the set-covering problem. *Operations Research*, 20(6):1152–1161.

Beasley, J. (1990). A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37(1):151–164.

Beasley, J. E. (1987). An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93.

Beasley, J. E. and Jörnsten, K. (1992). Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300.

Bentz, C., Cornaz, D., and Ries, B. (2012). Packing and covering with linear programming: A survey. *European Journal of Operational Research*.

Berge, C. (1961). Färbung von graphen, deren sämtliche bzw. deren ungerade kreise starr sind. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 10(114).

Berge, C. (1972). Balanced matrices. *Mathematical Programming*, 2(1):19–31.

Bolotashvili, G., Kovalev, M., and Girlich, E. (1999). New facets of the linear ordering polytope. *SIAM Journal on Discrete Mathematics*, 12(3):326–336.

Borndörfer, R. (1998). *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, TU Berlin.

Borndörfer, R. and Heismann, O. (2012). The hypergraph assignment problem. Technical Report 12-14, ZIB.

Borndörfer, R., Reuther, M., Schlechte, T., and Weider, S. (2011). A Hypergraph Model for Railway Vehicle Rotation Planning. In Caprara, A. and Kontogiannis, S., editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2011)*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 146–155, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ZIB Report 11-36.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). *Assignment Problems*. Society for Industrial and Applied Mathematics.

Cambini, R., Gallo, G., and Scutellà, M. G. (1992). Minimum cost flows on hypergraphs. Technical report, Department of Computer Sciences, University of Pisa.

Cambini, R., Gallo, G., and Scutellà, M. G. (1997). Flows on hypergraphs. *Math. Program.*, 77:195–217.

Cánovas, L., Landete, M., and Marín, A. (2002). Facet obtaining procedures for set packing problems. *SIAM Journal on Discrete Mathematics*, 16(1):127–155.

Caprara, A., Toth, P., and Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4):353–371.

Chandra, B. and Halldórsson, M. M. (2001). Greedy local improvement and weighted set packing approximation. *J. Algorithms*, 39(2):223–240.

Christof, T. and Loebel, A. (2008). PORTA: Polyhedron representation transformation algorithm, version 1.4. `http://comopt.ifi.uni-heidelberg.de/software/PORTA/index.html`.

Christof, T. and Reinelt, G. (2001). Decomposition and parallelization techniques for enumerating the facets of combinatorial polytopes. *Int. J. Comput. Geometry Appl.*, 11(4):423–437.

Chudnovsky, M., Robertson, N., Seymour, P., and Thomas, R. (2006). The strong perfect graph theorem. *Annals of Mathematics*, pages 51–229.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235.

Coppersmith, D. and Sorkin, G. B. (1999). Constructive bounds and exact expectations for the random assignment problem. *Random Structures & Algorithms*, 15(2):113–144.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1989). *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.

Cygan, M., Grandoni, F., and Mastrolilli, M. (2013). How to sell hyperedges: The hypermatching assignment problem. In *SODA*, pages 342–351.

De Vries, S. and Vohra, R. V. (2003). Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309.

Dobzinski, S. and Schapira, M. (2006). An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1064–1073.

Edmonds, J. (1965a). Maximum matching and a polyhedron with 0, 1 vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130.

Edmonds, J. (1965b). Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467.

Feige, U. (1995). A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*.

Feige, U. and Vondrak, J. (2006). Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 667–676. IEEE.

Fulkerson, D., Hoffman, A., and Oppenheim, R. (1974). On balanced matrices. In Balinski, M., editor, *Pivoting and Extension*, volume 1 of *Mathematical Programming Studies*, pages 120–132. Springer Berlin Heidelberg.

Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1987). A Schur-complement method for sparse quadratic programming. Technical report, Stanford University, Systems Optimization Lab.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer.

Hall, P. (1935). On representatives of subsets. *Journal of the London Mathematical Society*, 1(1):26–30.

Håstad, J. (1996). Clique is hard to approximate within $n^{1-\epsilon}$. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 627–636. IEEE.

Heismann, O. (2010). Minimum cost hyperassignments. Master's thesis, TU Berlin.

Heismann, O. and Borndörfer, R. (2012). Minimum Cost Hyperassignments with Applications to ICE/IC Rotation Planning. In Klatte, D., Lüthi, H.-J., and Schmedders, K., editors, *Operations Research Proceedings 2011*, pages 59–64. Springer Verlag. ZIB Report 11-46.

Heismann, O. and Borndörfer, R. (2013a). A generalization of odd set inequalities for the set packing problem. Accepted on December 27, 2013 for the post-conference proceedings of the OR 2013 Conference.

Heismann, O. and Borndörfer, R. (2013b). The random hypergraph assignment problem. In *Proceedings of the 16th International Multiconference INFORMATION SOCIETY – IS 2013*.

Heismann, O., Hildenbrandt, A., Silvestri, F., Reinelt, G., and Borndörfer, R. (2013). HUHFA: A framework for facet classification. Technical Report 13-45, ZIB.

Hildenbrandt, A., Reinelt, G., and Heismann, O. (2013). Integer programming models for the target visitation problem. In *Proceedings of the 16th International Multiconference INFORMATION SOCIETY – IS 2013*.

Hoffman, K. L. and Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682.

Jeroslow, R. G., Martin, K., Rardin, R. L., and Wang, J. (1992). Gainfree Leontief substitution flow problems. *Math. Program.*, 57(3):375–414.

Jerrum, M. (1986). A compact representation for permutation groups. *Journal of Algorithms*, 7(1):60–78.

Johnson, N., Kotz, S., and Balakrishnan, N. (1995). *Continuous univariate distributions*. Number 2 in Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley & Sons.

Kann, V. (1991). Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35.

Klabjan, D., Johnson, E. L., Nemhauser, G. L., Gelman, E., and Ramaswamy, S. (2001). Airline crew scheduling with regularity. *Transportation Science*, 35(4):359–374.

Koster, A. M. C. A., Zymolka, A., and Kutschka, M. (2009). Algorithms to separate $\{0, \frac{1}{2}\}$-chvátal-gomory cuts. *Algorithmica*, 55(2):375–391.

Krokhmal, P. A. and Pardalos, P. M. (2009). Random assignment problems. *European Journal of Operational Research*, 194(1):1–17.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Laurent, M. (2001). A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496.

Lehman, A. (1979). On the width-length inequality. *Mathematical Programming*, 16(1):245–259.

Linderothy, J. T., Leey, E. K., and Savelsbergh, M. W. P. (1999). A parallel, linear programming based heuristic for large scale set partitioning problems.

Linusson, S. and Wästlund, J. (2004). A proof of Parisi's conjecture on the random assignment problem. *Probability Theory and Related Fields*, 128(3):419–440.

Lovász, L. (1972). Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267.

Lovász, L. and Plummer, M. (1986). *Matching theory*. North-Holland mathematics studies. Akadémiai Kiadó.

Maróti, G. (2006). *Operations research models for railway rolling stock planning*. PhD thesis, Technische Universiteit Eindhoven.

McDermid, E. J. and Manlove, D. F. (2010). Keeping partners together: algorithmic results for the hospitals/residents problem with couples. *Journal of Combinatorial Optimization*, 19(3):279–303.

Mézard, M. and Parisi, G. (1985). Replicas and optimization. *Journal de Physique Lettres*, 46:771–778.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38.

Naddef, D. and Rinaldi, G. (1993). The graphical relaxation: A new framework for the symmetric traveling salesman polytope. *Mathematical Programming*, 58(1-3):53–88.

Orlin, J. B. (1997). A polynomial time primal network simplex algorithm for minimum cost flows. *Math. Program.*, 77:109–129.

Padberg, M. W. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215.

Padberg, M. W. and Rao, M. R. (1982). Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80.

Pêcher, A. and Wagler, A. (2006). Generalized clique family inequalities for claw-free graphs. *Electronic Notes in Discrete Mathematics*, 25:117–121.

Peng, Y. and Sissokho, P. A. (2013). The feasible matching problem. *Graphs and Combinatorics*, 29(3):695–704.

Rebennack, S. (2009). Stable set problem: Branch & cut algorithms. In *Encyclopedia of Optimization*, pages 3676–3688. Springer.

Rehn, T. (2010). Fundamental permutation group algorithms for symmetry computation. Diploma thesis, Otto von Guericke University Magdeburg. `http://fma2.math.uni-magdeburg.de/~latgeo/permlib/diploma-thesis-cs-rehn.pdf`.

Rehn, T. and Schürmann, A. (2010). C++ tools for exploiting polyhedral symmetries. In Fukuda, K., Hoeven, J., Joswig, M., and Takayama, N., editors, *Mathematical Software – ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 295–298. Springer Berlin Heidelberg.

Reinelt, G. (1985). *The linear ordering problem: algorithms and applications*, volume 8. Heldermann Berlin.

Rönnberg, E. and Larsson, T. (2009). Column generation in the integral simplex method. *European Journal of Operational Research*, 192(1):333–342.

Saket, R. and Sviridenko, M. (2012). New and improved bounds for the minimum set cover problem. In *APPROX-RANDOM*, pages 288–300.

Saxena, A. (2004a). On the set covering polytope: I. all the facets with coefficients in $\{0, 1, 2, 3\}$. Technical report, Tepper School of Business, Carnegie Mellon University.

Saxena, A. (2004b). On the set covering polytope: Ii. lifting facets with coefficients in $\{0, 1, 2, 3\}$. Technical report, Tepper School of Business, Carnegie Mellon University.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and combinatorics. Springer.

Shahrokhi, F. (2009). *Survey of approximation algorithms for set cover problem*. PhD thesis, University of North Texas.

Wedelin, D. (1995). An algorithm for large scale 0–1 integer programming with application to airline crew scheduling. *Annals of Operations Research*, 57(1):283–301.

Weyl, H. (1934). Elementare Theorie der konvexen Polyeder. *Commentarii Mathematici Helvetici*, 7(1):290–306.

Ziegler, G. M. (1995). *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer New York.

Index

Bold numbers indicate pages on which the keywords are introduced.